DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Multi-Object Tracking in Aerial and Satellite Imagery

Maximilian Kraus

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Multi-Object Tracking in Aerial and Satellite Imagery

Multi-Object Tracking in Luft- und Satellitenbildern

Author:	Maximilian Kraus
Supervisor:	Prof. Dr. Alois Knoll
Advisor:	Emeç Erçelik
Submission Date:	15th May 2020

I confirm that this master's thesis in informatics: robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, 7th May 2020 Maximilian Kraus

Acknowledgments

I would like to thank Reza Bahmanyar and Majid Azimi sincerely for their continuous support and advice from DLR's side. Additionally, I would like to thank Emeç Erçelik from TUM's side for accepting the supervisor role and giving valuable input whenever necessary.

Last but not least, I am grateful to my family and my friends who have supported me throughout my studies in every situation. Without them, I would not be where I am today.

Abstract

Multi-pedestrian and -vehicle tracking in aerial imagery has several critical applications, including event monitoring, disaster management, predictive traffic, and transport efficiency. While some research works already studied vehicle tracking in remote sensing scenarios, pedestrian tracking has not found the necessary attraction caused by the insufficient level of detail in aerial imagery. Recently, the development of better camera systems and the possibility to capture aerial imagery at low-cost paved the way for establishing novel tracking approaches based on remote sensing. However, current state-of-the-art algorithms, including deep learning based methods, perform especially poorly with pedestrians in aerial imagery, incapable of handling severe challenges such as the large number and the tiny size of the pedestrians (e.g., 4×4 pixels) with their similar appearances as well as different scales, atmospheric conditions, low frame rates, and moving camera. In contrast to vehicles moving along predetermined paths, such as highways or streets, pedestrians show more difficult motion characteristics posing additional demands on the tracker.

Within the scope of this master thesis, we propose AerialMPTNet, a novel regression-based deep neural network able to tackle the challenges of pedestrian and vehicle tracking in geo-referenced aerial imagery. AerialMPTNet fuses appearance features by a Siamese Neural Network with movement prediction of a Long Short-Term Memory and adjacent graphical features of Graph Convolutional Neural Network. In contrast to previous works, we encode the motion model and the adjacent neighbor modeling in an end-to-end fashion as part of the neural network. Consequently, our network can learn motion characteristics directly from the data and additionally learns to weight the influence of surrounding objects. Furthermore, to the best of our knowledge, we are the first to apply Squeeze-and-Excitation layers and Online Hard Example Mining to a regression-based deep tracker.

We evaluate AerialMPNet intensively on two aerial pedestrian datasets, AerialMPT and KIT AIS pedestrian. Both datasets consist of multiple image sequences captured at two frames per second on different flying altitudes, showing different crowd densities and different terrain (e.g., open-air concerts, Munich city areas, BAUMA trade fair). Results indicate that AerialMPTNet outperforms state-of-the-art algorithms such as Tracktor++, SMSOT-CNN or DCFNet on pedestrian tracking in aerial imagery significantly. Compared to the SMSOT-CNN baseline, multiple-object tracker accuracy (MOTA) improves by 18.8 points to -16.2 and by 13.8 points to -23.4 on KIT AIS pedestrian and AerialMPT, respectively.

Additionally, we evaluate our approach on the KIT AIS vehicle dataset. AerialMPTNet achieves a competitive MOTA score of 42.0. Since we fitted our method gradually for pedestrian tracking, other trackers achieve better scores here.

Contents

1 Introduction

Visual Object Tracking (VOT) is a fundamental challenge in computer vision dealing with locating objects in visual data over time. Although much progress has been made in recent years [1, 2, 3, 4], challenging problems such as heavy occlusions, different scales, background clutter or crowded scenes remain unsolved. VOT can be divided into single- and multiple-object trackers (SOTs and MOTs). While SOTs track one single object within a video or an image sequence, MOTs locate multiple objects at each frame. MOT scenarios are often more complex than SOT scenarios since trackers must handle a larger amount of objects in a reasonable time.

MOT methods using traditional approaches such as Kalman and particle filters [5, 6], Discriminative Correlation Filter (DCF) [7] or silhouette tracking [8] perform poorly in unconstrained environments. They are handicapped by handcrafted target representations (Histogram of Gradients (HOG) [9], color, position) and nondynamic target modeling [10]. In general, most of these works consider several constraints to simplify the tracking task significantly. Such constraints include, for example, stationary cameras, the limited amount of occlusion as well as the number of objects, and no sudden background or object appearance changes. The recent success of deep learning (DL) based techniques in object detection, segmentation and classification [11, 12, 13] influenced VOT significantly, leading to the development of better tracking methods based on Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNNs) [14, 15], Recurrent Neural Networks (RNNs) [16], Siamese Neural Networks (SNNs) [17, 18], Generative Adversarial Networks (GANs) [19] and custom architectures [20].

As of today, tracking algorithms play a crucial role in many fields. Critical applications for MOT in ground imagery are, for example, in autonomous driving, where tracking of nearby vehicles and pedestrians is required for path planning, or in visual surveillance, where tracking of suspicious pedestrians in CCTV footage is required for security reasons. Additionally, video-based pedestrian tracking and counting can provide unbiased information for tourism, public transportation, security, and safety. Tracking algorithms also have use cases in fields such as sports [21] (e.g., tracking of the ball in tennis matches), medicine [22] (e.g., path planning in computer-aided surgery), or biology [23] (e.g., evaluation of bacteria growth and movement).

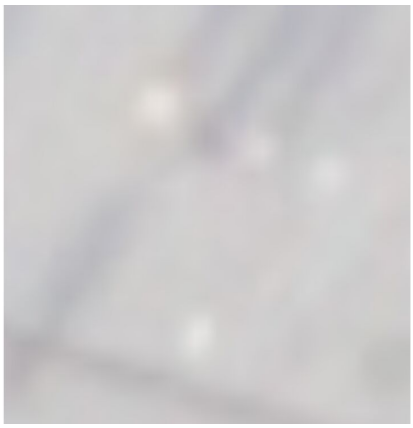
In the field of remote sensing, VOT was previously challenging to exploit due to the limited level of detail in aerial imagery. NASA's first generations of the Landsat satellites provided a resolution of 60m per pixel, for example. Such a resolution is not sufficient for VOT approaches dealing with small objects. Nevertheless, remote sensing provides a cheap way of collecting large-scale aerial imagery in a short amount of time. Recently, the development of better camera systems made very high-resolution aerial imagery available, opening the

field for a various number of applications ranging from the analysis of ecological systems to drone surveillance [24, 25]. Furthermore, aerial imagery provided by airborne platforms, such as helicopters and aircraft, is useful in several fields. In forestry, airborne images are useful to monitor tree yield, tree trimming, and fire. Additionally, the data allows the study and management of coastal ecosystems and can be used by insurances to monitor flooding [26, 27]. Current commercial satellites provide image data with resolutions of up to 30cm per pixel and allow high acquisition speeds. However, airborne platforms can deliver image data with even higher resolutions since the distance to the ground is lower than those of satellites. These developments alleviate previous limitations and make it possible to develop novel MOT methods dealing with aerial imagery and small objects such as ships, cars, and pedestrians, offering new chances in predictive traffic, event monitoring, transport efficiency, and security. Tracking and detection of humans also have use cases in disaster management. A good example is the recent coronavirus crisis, where such methods are valuable by giving authorities notice about people not following curfew rules.

Despite its practical applications, only a few research works have dealt with MOT scenarios in aerial imagery [28, 29, 30]. The vast amount of moving objects, multiple scales, and the small size of objects compared to standard computer vision scenarios makes tracking in aerial imagery especially difficult. Challenges also include low frame rates, different kinds of visibilities and weather conditions, large images, and moving cameras. Image data from drone or ground surveillance datasets, which find usage in standard MOT benchmarks such as MOT16 and MOT17 [31], differ significantly: Targets are bigger, placed with less spatial density and often have unambiguous appearance features discriminating them from other objects. Additionally, videos are recorded with higher frame rates and better quality.

In this master thesis, we aim to develop a deep learning based tracking method dealing with multi-pedestrian and multi-vehicle tracking in geo-referenced aerial image sequences. The image sequences were captured by an airborne platform during different flight campaigns of the German Aerospace Center (DLR) and vary significantly in crowd density, movement patterns, quality, and image size. The Ground Sampling Distance (GSD), which reflects the spatial size of a pixel in cm, varies between 8 cm and 13 cm. Consequently, the images were captured at different flight altitudes, and dependent on the altitude, similar objects can have different relative sizes in the image plane. The total number of objects per sequence ranges up to 609, and hence, hard memory and computational demands are placed on the tracker. Pedestrians occur in these images just as small points, hardly exceeding more than 4 x 4 pixels. Even for human experts, distinguishing multiple pedestrians based on their appearance is laborious and challenging. Vehicles appear as bigger objects and are easier to distinguish. However, different vehicle sizes, fast movements paired with low frame rates, and occlusions (e.g., when vehicles are located under bridges or partly occluded by trees or other vehicles) add to the complexity of this situation. Figure 1.1 illustrates the challenges of MOT in aerial imagery clearly.

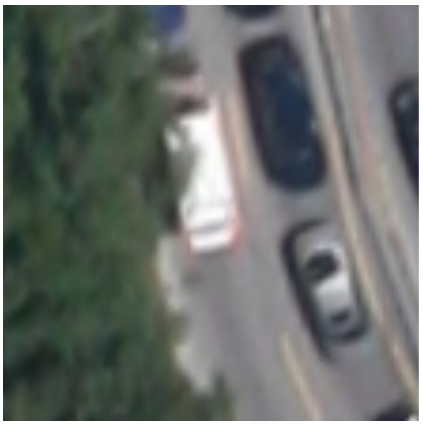
Generally, DNN-based tracking methods can be categorized into detection- and regression-based methods. Regression-based trackers usually receive two image samples, a target crop and a search crop, and directly regress an object's position based on the known initial position



(a) Crop showing multiple pedestrians. It is difficult to distinguish between them (here with shadows) and other similar-looking objects (here without shadows). The low contrast also poses severe challenges to a tracker.

(b) Crop showing multiple pedestrians at the BAUMA trade fair. The tracking of pedestrians in a small alley together with occlusions, shadows, and strong background colors is demanding.

(c) Crop showing multiple cars. The shadow on the right hand side poses a severe challenge to a tracker.



(d) Crop showing multiple cars. Vehicles on the left hand side are party occluded by trees.

Figure 1.1: Visualizations of some major challenges in aerial multi-object tracking. The crops are taken from the KIT AIS pedestrian (top-left), the AerialMPT (top-right) and the KIT AIS vehicle dataset (bottom).

in the target crop [32, 33]. Detection-based trackers mostly employ object detectors and link detections with the previous tracks to find a suitable trajectory. However, the current DL-based trackers are not well-equipped to handle the challenges of the scenarios we are facing. Due to the different atmospheric conditions, small objects are hardly visible in some samples, leading to a high amount of false-negatives and -positives in tracking-by-detection based frameworks. In contrast, regression-based trackers lack integrating essential information such as the previous movements of the targets.

Dealing successfully with the challenges of MOT in aerial imagery, we propose AerialMPTNet, a regression-based neural network improving the tracking performance for pedestrians and vehicles by a large margin compared to previous works. AerialMPTNet consists of an SNN which takes two image crops as input, namely a target and a search crop where the target position is known and has to be determined, respectively. The image crops contain appearance information that AerialMPTNet uses to regress an object's position. Furthermore, it is designed and trained to incorporate temporal and adjacent information of the movements of all objects in an end-to-end manner. Our method benefits from a Long Short-Term Memory (LSTM) for movement prediction and a GraphCNN for adjacent modeling (i.e., the spatial and temporal relationships between adjacent objects). The final output of the network is four coordinates describing the top-left and bottom-right corner of the object's bounding box in search crop coordinates. Additionally, to the best of our knowledge, we are the first work to apply Online Hard Example Mining (OHEM) and adaptive weighting of convolutional channels by using Squeeze-and-Excitation (SE) layers to a regression-based DL tracker. We evaluate the proposed network intensively on the KIT AIS[1] pedestrian and vehicle datasets and compare them with several state-of-the-art trackers. Additionally, we benchmark the proposed solution to the newly introduced Aerial Multi-Pedestrian Tracking (AerialMPT) dataset. The related paper is currently under review for the *ICPR2020* conference, and hence, we cannot give a reference here.

For pedestrian tracking, AerialMPTNet outperforms all previous methods in terms of MOTA (-16.2 and -23.4 for KIT AIS and AerialMPT, respectively). It reaches a competitive MOTP of 69.6 and 69.7. Adding the LSTM and GraphCNN modules bring consecutive performance gains. For vehicle tracking, AerialMPTNet reaches a competitive MOTA and MOTP of 42.0 and 76.3. Our methods, including the LSTM- and GraphCNN-based AerialMPTNet, rank within the three best-performing trackers.

In the following, chapter 2 provides an overview of related work and the necessary theoretical background. Afterwards, we introduce essential metrics and datasets in chapter 3. In chapter 4, we describe the preliminary experiments we conducted during the progress of this thesis. We explain our methodology in chapter 5 in detail and evaluate and discuss our proposed methods in chapter 6. Last but not least, we conclude this master thesis with chapter 7 and give some ideas for future work.

[1] https://www.ipf.kit.edu/code.php

2 Theoretical Background & Related Work

This chapter introduces the essential theoretical background and related works. We start by giving an overview of computer vision and remote sensing and show the distinction between them. Afterwards, we explain the basics of Machine and Deep Learning related to this thesis. This chapter concludes with background and related work of visual object tracking and its application in aerial imagery.

2.1 Computer Vision & Remote Sensing

Computer vision and remote sensing are related topics, both dealing with the processing and computer understanding of images. The goal of traditional computer vision is to make a computer autonomously perform some of the tasks the human visual system can perform and to infer something about the environment in which a picture was taken [34]. In contrast to pure image processing, the output of a computer vision algorithm gives back information about the given image, and not of a new picture. Information can be anything such as object detection and recognition results, camera position, 3D models, and image segmentation. Although computer vision tasks often seem trivial for humans, they remain highly complex for computers since visual perception is not bound to any specific environment, and since different types and amounts of occlusions can limit the view on a target. Additionally, a target can move, and hence, can be seen from different views and with different lighting conditions [35]. Nevertheless, there has been much progress in the field, especially with the rise of machine learning and deep learning, the affordable and easy access to computing power and the availability of mobile technologies offering massive amounts of photo and video data. Recently, computers surpassed the reported human-level-performance on the ImageNet dataset [36] for object classification [37]. Optical Character Recognition (OCR), medical imaging, machine inspection, facial recognition, pattern detection, surveillance, motion capturing, and feature matching are other fields of application for computer vision [35].

In contrast to computer vision, remote sensing deals with different scenarios: Observing the earth and the environment from space or very high altitudes and retrieving information from these observations. During the fast development of remote sensing, it provided us with a better understanding of weather and climate, leading to a more precise weather forecast, offers a cheap and effective way of collecting information of vast spatial regions, and provides methods to monitor ground objects over time [38, 24]. Such information is useful to analyze the development of rural and urban areas or the evolution of agricultural processes by providing repetitive knowledge on crop status during different times of a season.

These are only few examples. Others also include atmospheric research, data collection with different scales, resolutions, and sensors, analysis of ecological systems, mapping of wildfires and natural resources, and coordination of emergency responses [24, 25]. In recent years, the development of better camera systems made very high-resolution image ground data available. The higher level of detail opened the field for new areas of application, especially for remote sensing tracking approaches dealing with small objects such as cars, ships, or pedestrians [28, 29, 30].

2.2 Machine Learning

This section gives an overview of machine learning and introduces deep learning. Machine learning pursues to provide knowledge to computers through data and observations, which allows the computer to generalize to new situations. Machine learning tasks are divided into three categories: supervised learning, unsupervised learning, and reinforcement learning [39].

In supervised learning, a function maps inputs to specific outputs. Such inputs are provided as datasets $D = (x_1, x_2, ..)$, where x_i can be almost any kind of data, such as images, video files, sound files, point clouds, or time-series data [40]. Each sample x_i is paired with a desired output value y_i, which is called the ground truth. Afterwards, an algorithm can learn a function by analyzing the data and map unseen samples correctly. The ground truth can either be a discrete label or a continuous variable, dividing supervised learning into classification and regression tasks. Common datasets for classification are the MNIST dataset [41] containing handwritten digits, CIFAR-10 [42] which consists of more than 60,000 images for image classification as well as MS-COCO [43] containing images for object detection and segmentation. Popular regression datasets cover, for example, house price prediction given the details of houses and their neighborhood as well as predicting the quality of wine given specific attributes [44, 45].

The goal of unsupervised learning is to find previously unknown patterns or learn the underlying distribution of data given a dataset $D = (x_1, x_2, ..)$. In contrast to supervised learning, the samples x_i are not paired with an output value y_i. Instead, unsupervised learning aims to cluster similar samples within the data, perform density estimation, or solve association problems, for example.

Reinforcement learning mainly optimizes the actions of software agents in a given environment. The agent has no knowledge of which actions to take until it has been into a specific situation. Based on its own decisions, it receives a reward depending on the outcome of its action. Future decisions are affected by this reward. The final goal is to maximize the reward in order to maximize the agent's performance.

This thesis includes approaches based on supervised learning and deep learning. Hence, we will deepen the basics of these in the next sections.

2.2.1 Deep Learning & Neural Networks

A neural network is a computer system that can learn and perform task-specific jobs without being explicitly programmed [46]. It consists of an input layer, one or multiple hidden layers, and an output layer. The input of such networks is a vector which is transformed by the hidden layers of the network. In each of these layers operate neurons, producing an output based on a received input. Initially, random weights are assigned to each connection between the neurons, and bias is assigned to each neuron. These weights and biases are updated during the training process of the network, leading to more accurate outputs. The overall process is called backpropagation [47]. The term deep learning emerged as networks got more and more layers, and hence got "deeper". Figure 2.1 shows an example network with one hidden layer.

However, for computer vision tasks dealing with image or video data, networks based on such fully connected layers do not scale well. For example, a single neuron in the first hidden layer with an input image of size $300 \times 300 \times 3$ (i.e. 300 width, 300 height, 3 color channels) has $300 \times 300 \times 3 = 270,000$ weights. CNNs deal with this problem by using a combination of convolutional, pooling, and fully connected layers. Convolutional layers consist of k filters of different sizes $W \times H$. The filters are slid across the input image and compute dot products between the entries in the filters and the entries in the input resulting in k 2D activation maps, which are stacked to obtain the output [48]. Pooling layers reduce the spatial size of the output to decrease the amount of computation and parameters in the network. Networks built upon such structures play a key role in this thesis.

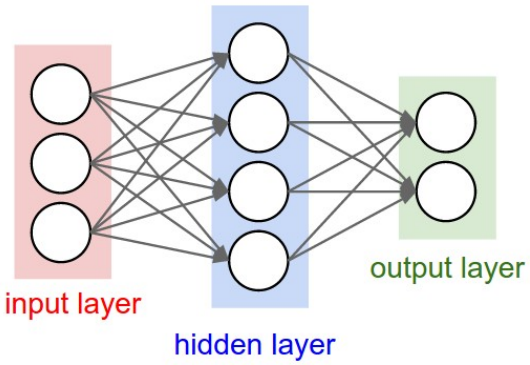


Figure 2.1: Neural network with hidden layers. Retrieved January 3, 2019: `http://cs231n.github.io/neural-networks-1/`

2.2.2 Layer Types

During the experiments of this thesis, we used various layer types. While explaining every type of layer in detail is beyond the scope of this thesis, we introduce unfamiliar ones in the following.

Local Response Normalization

Local Response Normalization (LRN) layers are not-trainable. They perform a lateral inhibition by applying a squared normalization over an input with multiple channels.

$$b_c = a_c \left(w + \frac{a}{n} \sum_{c'=max(0,c-n/2)}^{min(N-1,c+n/2)} a_{c'}^2 \right)^{-\beta} \tag{2.1}$$

, where a_c and b_c are the channel values before and after normalization, respectively, w is a value to provide numeric stability, n is the amount of neighboring channels used for normalization, a is a normalization constant and β is the exponent.

Squeeze-And-Excitation Layers

CNNs extract image information by sliding spatial filters across the input on different layer levels. While lower layers extract information such as edges and basic shapes, higher layers can detect more advanced structures such as cars or text. Nevertheless, each filter has a different relevance concerning the final output of the network. Within any layer, the filter amount is equal to the output depth. However, every output channel is weighted equally. SE Layers [49] change this behavior by weighting each channel adaptively while adding less than one percent of computing cost. Each channel is squeezed to a single value by using global average pooling [50], resulting in a vector with k entries. This vector is given to a fully connected layer reducing the size of the output vector by a certain ratio, followed by a Rectified Linear Unit (ReLU) activation function. The result is fed into a second fully connected layer scaling the vector back to its original size and applying a sigmoid activation afterwards. In a final step, the idea is to weight each channel of the original convolution block by multiplying the results of the SE block.

2.2.3 Activation Functions

The activation function calculates the output of a neuron based on its input. In order to make neural networks capable of approximating any arbitrary function, activation functions need to be non-linear functions [51]. Since neural networks need to be optimized during the training process, activation functions also need to be monotonic and differentiable. Commonly used activation functions are the sigmoid function, the hyperbolic tangent, or ReLU.

Sigmoid and tangent are recently losing popularity since they both suffer from the vanishing gradient problem. ReLU solves the problem of vanishing gradients by using a linear function with the positive x-axis. Nevertheless, since the output of all negative values is zero, neurons can "die" once they have a negative value. In order to solve this issue, ReLU was extended by Leaky ReLU, which has a predefined negative slope and PReLU, which aims to learn the slope of the activation function [52, 37]. Other activation functions include ELU, Maxout, and Swish, which was recently proposed by Google [53]. Figure 2.2 shows the functions and the graphs of some of the mentioned functions.

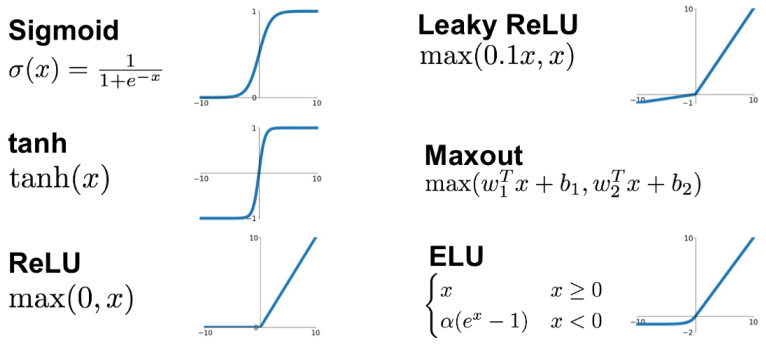


Figure 2.2: Activation Functions. Retrieved January 2, 2019: `https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044`

2.2.4 Loss Functions

Loss functions measure how well neural networks model the given data. They compare the output of the network with the ground truth and calculate the cost or the error. The loss function optimizes the network's weights w until an error threshold is met. It is also essential that the loss function is differentiable since neural networks update all parameters by applying backpropagation. This subsection introduces the losses relevant to this master thesis.

L1 & L2 Loss

The L1[1] and L2[2] losses are useful when dealing with regression problems. The L1 loss measures the Mean Absolute Error (MAE) between the output of the network x and the ground truth y. It is calculated as followed:

$$L1(x, y) = \sum_i |x_i - y_i| \tag{2.2}$$

The L1 loss is less affected by outliers than the L2 loss. The L2 loss calculates the error of the squared distance between the network output and the true value:

$$L2(x, y) = \sum_i (x_i - y_i)^2 \tag{2.3}$$

Huber Loss

The Huber loss [54] is a mixture of the L1 and the L2 losses and combines their good properties. It is given as the Mean Squared Error (MSE) when the error is small and calculated as the MAE when the error is big.

$$L_H(x, y) = \sum_i z_i \tag{2.4}$$

[1] https://pytorch.org/docs/stable/_modules/torch/nn/modules/loss.html#L1Loss
[2] https://pytorch.org/docs/stable/_modules/torch/nn/modules/loss.html#MSELoss

$$z_i = \begin{cases} 0.5(x_i - y_i)^2, & if|x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & otherwise \end{cases} \tag{2.5}$$

The Huber loss is more robust to outliers than the L2 loss and improves the L1 loss regarding missing minima at the end of the training.

2.3 Visual Object Tracking

VOT describes the process of locating one or multiple objects in video or image sequence data over time. The traditional tracking process consists of four phases: Initialization, appearance modeling, motion modeling, and object finding. During initialization, the targets are annotated by hand or an object detector. There exist different annotation styles; hence, regions can be annotated with points, bounding boxes, centroids, or object contours. In appearance modeling, recognizing visual features of the region of interest for better representation and the usage of various learning-based models to detect objects are important steps. Different scales, rotations, shifts, and occlusions make this a challenging problem. As of today, features have the most important role in tracking algorithms and can generally be divided into handcrafted and deep features. However, handcrafted features such as HOG, color names, or Scale-Invariant Feature Transform (SIFT) are outdated, and the research focus has recently shifted towards deep features, able to incorporate multi-level information and to be more tolerant against appearance variations [55]. Nevertheless, the use of DNNs is only possible if enough training data is available. The motion modeling step aims to predict the motion of the objects in future frames and gives back an estimate of the object's location. This procedure can effectively reduce the search space. Suitable methods for this step include Kalman filter [56], Sequential Monte Carlo methods [57] or RNNs. The last step includes finding the objects by linking objects close to the positions provided by the motion model with the available tracks and creating new tracks with unmatched objects. During the whole process, changing target and motion characteristics are integrated into the appearance and motion model. These steps can vary greatly depending on the framework, but give a general insight into the tracking process.

As stated previously, Trackers can be divided into SOTs [58, 59] and MOTs [60, 30]. SOTs only track a single object throughout the video, even if there are multiple objects visible in the frames. The object which one wants to track needs to be specified. MOTs can track multiple objects at the same time but can suffer from exponential runtime complexity increase and are slower compared to SOTs. Additionally, there exist detection-based [61] and detection-free trackers [62]. Detection-based methods use object detectors to find objects in each frame. Detection-free approaches mostly have a better runtime but are not able to detect new objects and require manual initialization. Another distinguishing feature is the learning strategy. Tracking algorithms trained with an online learning strategy can learn about the tracked object during runtime, giving it the ability to track generic objects [63]. Algorithms trained with an offline learning strategy do not learn during runtime, and hence offer a better runtime complexity [64]. Furthermore, tracking algorithms can be divided into online and offline

methods. Offline trackers take advantage of past and futures frames, while online trackers can only infer about past frames. Future frames can increase the tracking performance; however, in real-world scenarios these are not available. Most existing tracking approaches use a two-stage tracking-by-detection paradigm [65, 66]. In the first stage, a set of target samples is generated around the previously estimated position using region proposal, random sampling, or similar methods. In the second stage, each target sample is either classified as background or as a target object. In one-stage-tracking, the model receives a search sample together with a target sample as input and directly outputs a response map or coordinates through a learned regressor. The maximum value of the response map or the coordinates indicates the object position.

Traditional tracking methods mostly rely on the use of Kalman and particle filters to estimate an object's position. These methods use velocity and position information to perform tracking [5, 6, 67]. Tracking methods based solely on such approaches are outdated since they perform poorly in unconstrained environments. Nevertheless, such filters can predict and propagate object movements into the next frame, and as a result of this, minimize the search space. Other trackers follow a tracking-by-detection paradigm using template matching [68]. A chosen target patch models the appearance of a selected region of interest in the first frame. Matching regions can be found in a new frame using correlation, normalized cross-correlation, or the sum of squared distances [69, 70]. However, scale, illumination, and rotation changes are difficult to model with these methods. More advanced methods, also using the tracking-by-detection paradigm, rely on discriminative modeling. Discriminative classifiers separate the target from its background within a specific search space. There exist many different approaches, reaching from boosting methods to support vector machines [71, 72]. Other tracking algorithms utilize correlations filters for visual tracking. Correlation filters model the target's appearance by using filters trained on images. A target is initially selected based on a small crop of the first frame with the target centered in the middle. The tracking process works by convoluting the learned filter over the search window in the next frame. The output is a response map, the correlation output, with a peak representing the accurate target position. Afterwards, an online update renews the appearance information of the target. Since the correlation can be computed in the Fourier domain, such trackers achieve high frame rates. Examples of correlation-based trackers include MOSSE and KCF [7, 73]. Recently, the focus of researchers has shifted to DL-based tracking methods. The advantage of DL-based features over HOG, raw pixels values or grey-scale templates is enormous, and enable the modeling of appearance changes, occlusion situations, and dynamic environments effectively. The variety of proposed solutions is great: Methods include re-identification with appearance modeling and deep features [60], position regression mainly based on SNNs [18, 17], path prediction based on RNN-like networks [74] and object detection with DNNs such as YOLO [75].

In the following subsections, we aim to present an overview of VOT approaches relevant to this thesis. In subsection 2.3.1, we introduce the methods we adopted during the scope of this thesis. In subsection 2.3.2, we detail related work in the field of aerial object tracking.

2.3.1 Tracking Algorithms

Kalal et al. proposed Median Flow [76], which utilizes point and optical flow tracking. The input to the tracker is two consecutive images together with the initial bounding box of an object. The tracker calculates a set of points from a rectangular grid within the bounding box. Each of these points is tracked by a Lucas-Kanade tracker generating a sparse motion flow. Afterwards, the framework evaluates the quality of the predictions and filters 50 % of the worst. The remaining point predictions are used to calculate the new bounding box positions based on displacement.

MOSSE [7], KFC [73] and CSRT [77] are based upon DCFs. MOSSE got introduced by Bolme et al. in 2010. It uses a new type of correlation filter called Minimum Output Sum of Squared Errors (MOSSE), which aims to produce stable filters when initialized using only one frame and grey-scale templates. MOSSE is trained with a set of training images f_i and training outputs g_i, where g_i is generated from the ground truth as a Gaussian centered on the target. This framework achieved state-of-the-art performance while running with high frame rates. Henriques et al. [73] replaced the grey-scale templates with HOG features and proposed the idea of Kernelized Correlation Filter (KCF) in 2014. KCF works with multiple channel-like correlation filters. Additionally, the authors propose to use non-linear regression functions more powerful compared to linear functions and obtain non-linear filters that are trained and evaluated as fast as linear correlation filters. Similar to KCF, dual correlation filters use multiple channels. However, they are based on a linear kernel reducing computational complexity while maintaining almost the same performance compared to non-linear kernels. Recently, Lukezic et al. [77] proposed to use channel and reliability concepts to improve tracking based on DCFs. Channel-wise reliability-scores weight the influence of the learned filters and improve localization by reflecting the quality of the filters. Furthermore, a spatial reliability map attracts the filters mainly to the part of the object suitable for tacking, making it possible to widen the search space and improves the tracking scores of non-rectangular objects.

As we stated previously, the choice of appearance features plays a crucial role in object tracking. However, most previous work of DCF-based tracking approaches utilizes handcrafted features such as HOG, grey-scale features, raw pixels, and ColorNames or deep features trained independently on other tasks. Wang et al. [58] proposed an end-to-end trainable network architecture, able to learn convolutional features and perform the correlation-based tracking simultaneously. The authors encode the DCF as a correlation filter layer into the network, making it possible to backpropagate the weights trough it. Since the calculations remain in the Fourier domain, the run time complexity of the filter is preserved. The convolutional layers in front of the DCF encode prior tracking knowledge learned during an offline training process. The DCF defines the network output as the probability heatmap of object location.

In the case of generic object tracking, the learning strategy is typically entirely online. However, online training of neural networks is slow due to backpropagation – and this also leads to a high run time complexity. Held et al. [18] introduce a regression-based tracker called GOTURN based on a siamese neural network. Tracking methods able to track generic

objects usually require online training which is slow due to backpropagation; however, Held et al. can solve this problem by using an offline training method that makes the network understand a relationship between appearance and motion, and hence, the tracking process gets significantly faster. The knowledge gained during the offline training can be applied to track new unknown objects in an online manner. Without online backpropagation, GOTURN can track generic objects at 100 fps during test time. The input to the network consists of the two image tiles: One crop from a previous frame centered at the known object position and another crop from the current frame centered at the same position. The size of the crops is a factor of the object's bounding box size and is controllable by a hyperparameter, influencing the amount of context the network receives around the target object. The final output is the coordinates of the object in the current crop, which can be transformed in image coordinates. GOTURN achieves state-of-the-art performance on common SOT benchmarks such as VOT 2014[3]. However, all of the previously mentioned methods work with single objects only.

Bewley et al. [78] propose to use a simple multiple tracking approach based on the Jaccard distance, the Kalman Filter, and the Hungarian algorithm [79], a method that solves assignment problems globally in polynomial time. Bounding box position and size are the only measures to perform motion estimation and data association; the use of appearance features is neglected. In the first step, objects are detected using Faster R-CNN [13]. Afterwards, a linear constant velocity model approximate the movements of objects between consecutive frames independent of other objects. The algorithm compares detections to the predicted bounding boxes with an Intersection over Union (IoU) score and creates a cost matrix as a result. The Hungarian algorithm solves the assignment of detections to targets. The detections associated with the targets are used to update the target states via a Kalman filter framework. The state of unmatched targets is predicted without any state update. SORT runs with more than 250 Frames per Second (FPS) while achieving almost state-of-the-art accuracy. Occlusion scenarios and re-identification issues are ignored by the authors, making it not suitable for long-time tracking. Wojke et al. [60] extend SORT and tackle the occlusion and re-identification challenges formerly neglected in SORT. Track handling and the Kalman filtering module are almost identical to SORT. A significant difference lies in the assignment process. Wojke et al. use two additional metrics: Motion information provided by calculating the Mahalanobis distance between predicted bounding boxes and detections as well as appearance information by calculating the smallest cosine distance between the appearance features of a detection and the appearance features of an object already tracked. These features are calculated with a deep neural network trained on a large person re-identification dataset [80]. A cascade method determines object-to-track assignments - tracks which were updated recently are matched first while older tracks are matched in later stages. The cascade strategy effectively encodes the probability spread in the association likelihood. The framework falls back to SORT for cases where the cascade method could not match detections and targets.

Recently, Bergmann et al. [2] introduced a new tracking paradigm based on the object detector Faster R-CNN. Faster R-CNN uses object proposals that are given to object classification and a bounding box regression head of the neural network. The regression head tightens

[3]https://www.votchallenge.net/vot2014/

the bounding box to fit around the object. The authors train Faster R-CNN on the MOT17Det pedestrian dataset [31]. In the first step, Faster R-CNN performs object detection. The objects found in the first frame are afterwards initialized as tracks and tracked by regressing the bounding box with the regression head in the next frame while the identity of the objects is automatically preserved. Furthermore, Bergman et al. demonstrate that lost or deactivated tracks can be re-identified in the following frames using siamese networks together with a constant velocity motion model. Their approach called Tracktor++ can track multiple objects and consists of an uncomplicated framework.

2.3.2 Tracking in Satellite and High Resolution Aerial Imagery

VOT, especially of pedestrians and vehicles, in satellite and aerial imagery, is a challenge only a few previous works have tackled yet. Different scales, the vast amount of moving objects, and the tiny size of the objects (e.g., 4×4 pixels for pedestrians, 30×15 for vehicles) increase the difficulty and complexity of tracking in such scenarios significantly. The small object size compared with low contrast also leads to objects sharing similar appearance features. Further challenges include low frame rates, different kinds of visibilities, and weather conditions as well as moving cameras and huge resolutions. Data from standard aerial or ground surveillance datasets [81, 31] differs a lot since objects are bigger and have unique appearance features, videos are recorded with higher frame rates and objects are placed less densely.

Most approaches dealing with this kind of scenario are based on moving object detection [28, 29, 82]. One of the earliest approaches focusing on such wide-area scenarios, concentrating on vehicles driving on highways mainly, is made by Reilly et al. [28]. They eliminate camera motion by a point correspondence based correction method. Afterwards, they use a median background image modeled from ten frames and apply this median image for motion detection by subtracting it from the original frame. The results of this operation are the position of moving objects. All images are divided into overlapping grids. Each of these grids defines an independent tracking problem. Objects are tracked using bipartite graph matching between a set of label nodes and a set of target nodes, and the Hungarian Algorithm solves the cost matrix afterwards to receive the assignments. The usage of the grids makes it possible to track a large number of objects by reducing the runtime complexity of the Hungarian Algorithm to $O(n^3)$. Meng et al. [29] go along the same path. Their goal is to track objects such as ships and grounded aircraft. They detect moving objects by calculating an Accumulative Difference Image (ADI) from frame to frame. Pixels with high values in the ADI are likely to be moving objects. Each target is afterwards modeled by extracting its spectral and spatial features, where spectral features refer to the target's probability density function and the spatial features to the target's geometric area. Given the target model, matching candidates are found in the following frames via regional feature matching using a sliding window paradigm.

However, tracking methods based on moving object detection have severe disadvantages in our scenario. For instance, Reilly et al. use a road orientation estimate to constrain the assignment problem. Such an estimate may work for vehicles moving along predetermined paths such as highways and streets; however, pedestrians have more diverse and complex

movement behaviors, often resulting in crowded situations and multiple crossings. In general, such methods perform poorly in unconstrained environments, are sensitive to changing light and atmospheric conditions (e.g., clouds, shadows, or fog) and suffer from the parallax effect, not working well with small or static objects. Additionally, finding the moving object requires the usage of multiple frames, and hence these methods do not work in real-time.

Methods based on appearance-like features overcome these issues [83, 84, 85, 86, 30], making it possible to detect small and static objects on single images. While there is a huge amount of literature covering the topic of pedestrian and vehicle tracking in ground surveillance scenarios [14, 87], the amount of literature covering aerial and remote sensing scenarios is limited. Butenuth et al. [83] deal with pedestrian tracking in aerial image sequences. They apply an iterative Bayesian tracking approach, enabling the possibility to track many people. A pedestrian is described by position, color, and direction, and a linear dynamic model predicts futures states. Each link between a prediction and a detection is weighted by evaluating the state similarity and associated with the direct link method described in [61]. Schmidt et al. [84] introduce a tracking-by-detection framework based on Haar-like features. They use a Gentle AdaBoost classifier for object detection and an iterative Bayesian tracking approach, similar to [83]. Additionally, they calculate the optical flow between consecutive images to extract motion information. However, due to the difficulties of detecting small objects in aerial imagery, the regular occurrence of false positives and negatives influences the tracking performance negatively. Liu and Mattyus and Qui et al. [85, 86] aim to detect cars and ships in aerial imagery, respectively. However, their work is concentrated on object detection and not on visual tracking.

Bahmanyar et al. propose Stack of Multiple Single Object Tracking CNNs (SMSOT-CNN) [30] and extend the work of Held et al. by stacking the SOT GOTURN architecture in order to track multiple pedestrians and vehicles in aerial image sequences. SMSOT-CNN is the only previous work dealing with MOT in remote sensing by using DL. Bahmanyar et al. expand the network by three additional convolutional layers in front of the fully connected layers to improve the tracker's performance. In their architecture, each SOT-CNN is responsible for tracking one object solely; achieving MOT without exponential complexity increase when the number of objects is climbing. They evaluate their approach on the vehicle and pedestrian sets of the KIT AIS dataset[4] for object tracking in aerial image sequences. A comparison of the GOTURN network and SMSOT-CNN showed that MOTA improved by 16.4 points to 41.1 on the vehicle dataset and by 11.2 points to -29.8 on the pedestrian dataset. Nevertheless, SMSOT-CNN is dealing poorly with crowded situations and objects sharing similar appearance features, resulting in identity switches and loosing of tracks.

[4]https://www.ipf.kit.edu/code.php

3 Datasets & Metrics

3.1 Datasets

In this section, we introduce the datasets used in the scope of this thesis. Both tracking datasets, the KIT AIS[1] and the Aerial Multi-Pedestrian Tracking (AerialMPT) dataset, are provided by the German Aerospace Center (DLR) in order to reduce the lack of tracking and object detection datasets for aerial imagery and improve the possibilities to develop well-performing tracking methods. Additionally, we had access to the DLR's Aerial Crowd Dataset (DLR-ACD) [88], which consists of static aerial crowd scenes.

The capturing process for both tracking datasets was similar. All images were taken by the DLR's 3K camera system, containing a nadir-looking and two side-looking DSLR cameras, mounted on an airborne platform flying at different altitudes resulting in multiple spatial resolutions, so-called GSDs. The camera is continuously moving; hence, in a post-processing step, all images were orthorectified with a digital elevation model, co-registered, and geo-referenced with a GPS/IMU system. Afterwards, images taken at the same time are fused into one single image and cropped to the region of interest. This overall process leads to small errors visible in the frame alignment. The frame rate of all sequences is 2 Hz. The image sequences were captured during different flight campaigns of DLR and differ significantly in crowd density, movement patterns, quality, image size, viewing angle, and terrain, where the number of frames per sequence is dependant on the overlap in flight direction and camera configuration. In the following, we introduce the KIT AIS and the AerialMPT datasets in detail. Additionally, we present the DLR-ACD shortly.

3.1.1 KIT AIS

The KIT AIS dataset is composed of two branches, vehicle tracking, and pedestrian tracking. For both branches, researchers and students labeled all images manually, leading to a small number of human errors. Vehicles are annotated by the smallest enclosing rectangle oriented in the direction of travel (i.e., bounding box), individual pedestrians with point annotations centered on the person's head. However, for tracking purposes, we also used bounding boxes with sizes of 4×4 pixels and 5×5 pixels dependant on the GSD of the pedestrian sequences. Objects can vanish and disappear (e.g., leaving the scene, occluded by another object), and consequently, tracks do not have to be labeled continuously. In the vehicle branch, cars, trucks, and busses are labeled; bicycles, motorcycles, cable cars, shadows of vehicles, and vehicles, which are partly outside of the image region or covered by more than $\frac{1}{3}$ of their size are not.

[1] https://www.ipf.kit.edu/code.php

Table 3.1: Statistics of the training and testing set of the KIT AIS pedestrian dataset. The image sequences are from different flight campaigns over **Alianz Arena forecourt** (Munich, Germany; named AA_*), **OAC** Open-Air Concert (Germany; named RaR_*) and **Karlsplatz** (Munich, Germany; named Munich*).

Training

Seq.	Resolution	nFrames	nPedestrian	nAnno.	nAnno./fr.	GSD (cm)
AA_Crossing_01	309 x 487	18	164	2,618	145.44	15.0
AA_Easy_01	161 x 168	14	8	112	8.00	15.0
AA_Easy_02	338 x 507	12	16	185	15.42	15.0
AA_Easy_Entrance	165 x 125	19	83	1,105	58.16	15.0
AA_Walking_01	227 x 297	13	40	445	34.23	15.0
Munich01	509 x 579	24	100	1,308	54.5	12.0
RaR_Snack_Zone_01	443 x 535	4	237	930	232.5	15.0
Total		104	633	6,703	64.45	

Testing

Seq.	Resolution	nFrames	nPedestrian	nAnno.	nAnno./fr.	GSD (cm)
AA_Crossing_02	322 x 537	13	94	1,135	87.31	15.0
AA_Entrance_01	835 x 798	16	973	14,031	876.94	15.0
AA_Walking_02	516 x 445	17	188	2,671	157.1	15.0
Munich02	702 x 790	31	230	6,125	197.58	12.0
RaR_Snack_Zone_02	509 x 474	4	220	865	216.25	15.0
RaR_Snack_Zone_04	669 x 542	4	311	1,230	307.50	15.0
Total		85	2016	26,057	306.55	

In the pedestrian branch, only pedestrians are labeled. Due to crowded scenarios or adverse atmospheric conditions, pedestrians can be hardly visible, and tracks are estimated as precise as possible in these situations. The GSDs of all sequences range from 12 to 15 cm.

KIT AIS pedestrian consists of 13 sequences with a total of 32,760 pedestrian annotations, including seven training and six testing sequences with 104 and 85 frames, respectively. The length of all sequences differs between 4 and 31 frames, with the mean hovering at 14.5 frames. The images were recorded from Munich's city center as well as from mass event in front of Allianz Arena (Munich) or open-air concerts (OAC). Table 3.1 gives an overview of the statistics of this branch.

KIT AIS vehicle composes nine tracking sequences with 239 frames. The vehicle branch has no pre-defined training/testing sets. Hence, we split it into a training and testing set with 5 and 4 sequences, respectively. In total, the training and testing set consist of 131 and 108 frames. The length of all sequences varies between 14 to 47 frames, with the number of vehicles ranging from 16 to 88. Similar to KIT AIS pedestrian, we present the statistics of this subset in Table 3.2. The dataset includes several difficult tracking challenges, such as overtaking and turning maneuvers, lane changes as well as partial and total occlusion by big objects such as bridges.

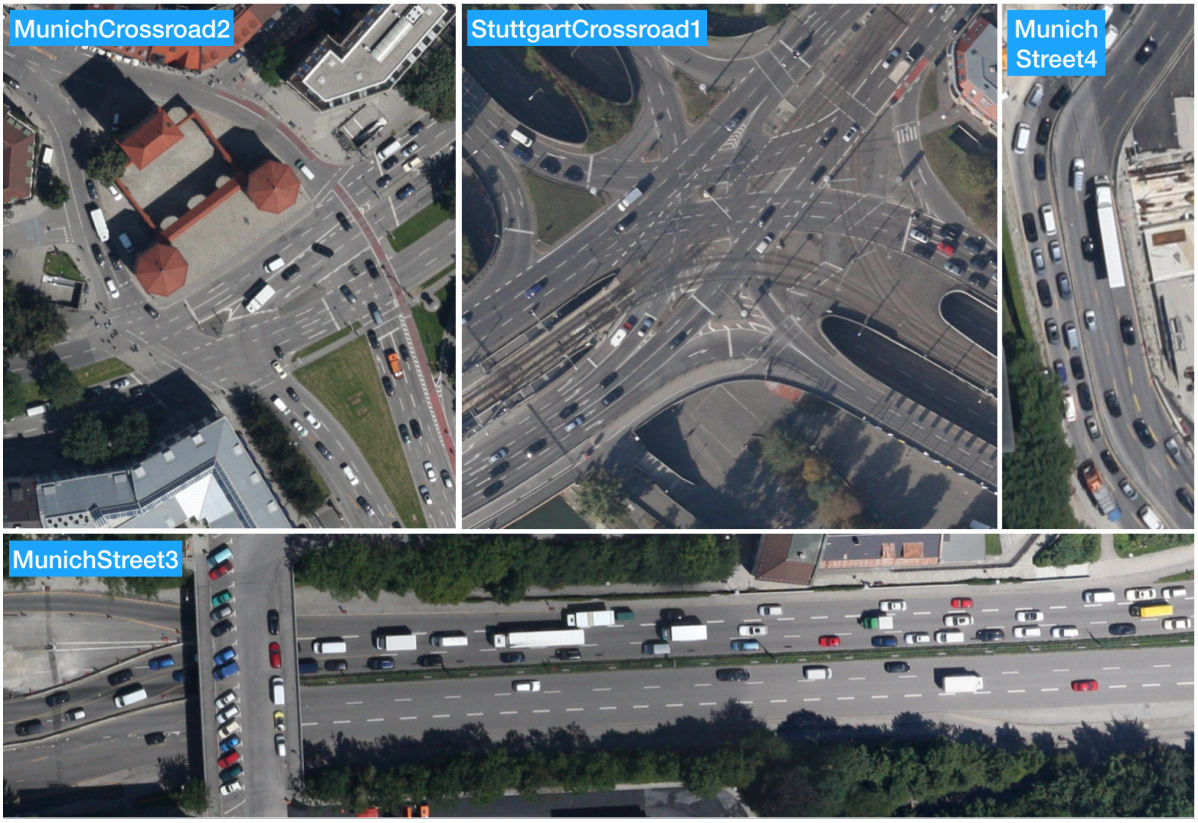


Figure 3.1: Sample images from the KIT AIS vehicle dataset captured at different locations in Munich and Stuttgart, Germany. The pictures show varying spatial densities and give an overview of the dataset.

Table 3.2: Statistics of the training and testing sets of the KIT AIS vehicle dataset. The image sequences are from different flight campaigns over **highways**, **crossroads** and **streets** in Munich and Stuttgart.

Training

Seq.	Resolution	nFrames	nVehicles	nAnno.	nAnno./fr.	GSD (cm)
MunichAutobahn1	633 x 988	16	16	161	10.06	15.0
MunichCrossroad1	684 x 547	20	30	509	25.45	12.0
MunichStreet1	1,764 x 430	25	57	1,338	53.52	12.0
MunichStreet3	1,771 x 422	47	88	3,071	65.34	12.0
StuttgartAutobahn1	767 x 669	23	43	764	33.22	17.0
Total		131	234	5,843	44.60	

Testing

Seq.	Resolution	nFrames	nVehicles	nAnno.	nAnno./fr.	GSD (cm)
MunichCrossroad2	895 x 1,036	45	66	2,155	47.89	12.0
MunichStreet2	1,284 x 377	20	47	746	37.30	12.0
MunichStreet4	1,284 x 388	29	68	1,519	52.38	12.0
StuttgartCrossroad1	724 x 708	14	49	554	39.57	17.0
Total		108	230	4,974	46.06	

3.1.2 AerialMPT

The AerialMPT (Aerial Multi-Pedestrian Tracking) dataset is a new pedestrian tracking dataset, currently under review to be published in ICPR 2020 conference. The access to this dataset will be public, enabling the promotion of research on aerial MOT. AerialMPT is dealing with the limitations of the KIT AIS dataset, such as low image quality and low degree of diversity. It consists of 14 sequences and a total of 307 frames. Images were acquired by a newer version of the DLR 3K camera system, resulting in images with better contrast compared to the KIT AIS dataset.

Similar to KIT AIS pedestrian, each person was manually labeled with point annotations centered on the individual's head by specialist staff and got assigned a unique ID over the whole sequence. Due to the similar appearance of adjacent pedestrians as well as other small objects, discriminating and rediscovering each person is time-consuming and difficult. The annotations were sanity checked as a part of this thesis. In total, AerialMPT comprises 2,528 pedestrians with 44,740 annotations points. The dataset composes a training and testing set with 8 and 6 image sequences, including 1,132 and 1,396 pedestrians on 179 and 128 frames, respectively. The length of all sequences varies between 8 and 30 frames. The image sequences were captured from different crowd scenarios, e.g., from moving pedestrians on mass events like OACs and fairs or from sparser scenarios in Munich's shopping streets. Table 3.3 details the statistics of this dataset. Additionally, we visualized the ground truth annotations for a sample image in Figure 3.2.

Table 3.3: Statistics of the training and testing set of the AerialMPT dataset. The image sequences are from different flight campaigns over **Bauma** construction trade fair (Munich, Germany), **OAC** Open-Air Concert (Germany), **Witt** Church day (Wittenberg, Germany), as well as **Pasing**, **Marienplatz**, and **Karlsplatz** Munich city areas (Germany).

Training

Seq.	Resolution	nFrames	nPedestrian	nAnno.	nAnno./fr.	GSD (cm)
Bauma1	462 x 306	19	270	4,448	234.11	11.5
Bauma2	310 x 249	29	148	3,627	125.07	11.5
Bauma4	281 x 243	22	127	2,399	109.05	11.5
Bauma5	281 x 243	17	94	1,410	82.94	11.5
Marienplatz	316 x 355	30	215	5,158	171.93	10.5
Pasing1L	614 x 366	28	100	2,327	83.11	10.5
Pasing1R	667 x 220	16	86	1,196	74.75	10.5
OAC	186 x 163	18	92	1,287	71.50	8.0
Total		179	1,132	21,852	122.08	

Testing

Seq.	Resolution	nFrames	nPedestrian	nAnno.	nAnno./fr.	GSD (cm)
Bauma3	611 x 552	16	609	8,788	549.25	11.5
Bauma6	310 x 249	26	270	5,314	204.38	11.5
Karlsplatz	283 x 275	27	146	3,374	124.96	10.0
Pasing7	667 x 220	24	103	2,064	86.00	10.5
Pasing8	614 x 366	27	83	1,932	71.56	10.5
Witt	353 x 1,202	8	185	1,416	177.00	13.0
Total		128	1,396	22,888	178.81	

Figure 3.2: Sample aerial image with its overlaid annotations from the AerialMPT dataset taken over the BAUMA 2016 trade fair. We annotated the pedestrians with different colors to make them easier to distinguish.

3.1.3 Comparison of KIT AIS pedestrian and AerialMPT datasets

Previously, the KIT AIS pedestrian dataset was the only dataset available for pedestrian tracking in aerial imagery. Besides its uniqueness, it deals with several drawbacks. The AerialMPT dataset was published to alleviate the limitations of the KIT AIS pedestrian dataset. It is striking that the number of minimum annotations per frame, as well as the number of total annotations of AerialMPT, is significantly higher than in the KIT AIS dataset. Based on visual inspection, sequences include pedestrians with more complex and realistic movement patterns as well as denser crowds compared to the KIT AIS pedestrian dataset. Additionally, all image sequences contain at least 50 persons, in contrast to the KIT AIS pedestrian dataset, where more than 20 % of the sequences include less than ten pedestrian tracks. The sequences in AerialMPT differ in weather conditions and visibility, incorporating more diverge kinds of shadows compared to the DLR KIT pedestrian dataset. Furthermore, the sequences of AerialMPT mostly contain a higher amount of frames: 60 % of the sequences consist of more than 20 frames compared to less than 20 % in KIT AIS pedestrian. We visualized some samples of both datasets in Figure 3.3. The image samples of KIT AIS have lower quality and contrast compared to AerialMPT.

3.1.4 DLR's Aerial Crowd Dataset

The DLR-ACD dataset [88] consists of 33 large aerial RGB images showing different mass events and urban scenes containing crowds, such as sports events, city centers, open-air fairs, and festivals. In contrast to KIT AIS and AerialMPT, which are tracking datasets, DLR-ACD is a pure crowd dataset and only consists of static images. These images were recorded with

Figure 3.3: Sample images from the KIT AIS pedestrian and the AerialMPT dataset. The samples annotated with Bauma3, OAC, Witt, and Pasing1 are included in AerialMPT, the remaining ones in KIT AIS. The images show that the contrast of image sequences in the AerialMPT dataset is significantly better compared to the KIT AIS pedestrian dataset.

a similar camera system as the previously presented dataset; however, the flying altitudes are ranging from 500 to 1600 m, resulting in GSDs varying between 4.5 and 15 cm and an average image size of 3619x5226 pixels.

Similar to AerialMPT, DLR-ACD was labeled manually with point-annotations resulting in 226,291 person annotations, ranging from 285 to 24,368 per image. However, most of the images contain a large number (>2K) of pedestrians, which is the main difference to other crowd datasets. The crowd density can vary significantly within a single image due to the massive field of view. We visualized some sample images in Figure 3.4.

3.2 Metrics

In this section, we introduce the most important metrics we use during the quantitative evaluation part of this thesis. We follow the work of Milan et al. [31] and report all of the widely used metrics in the MOT domain shown in Table 3.4.

The challenge of MOT is to find the spatial positions of p given objects given an image sequence, resulting in trajectories consisting of bounding boxes. A bounding box is defined by its x and y positions of the top-left and bottom-right corner in image coordinates together with its respective frame.

The quantification of a tracker's performance is dependant on whether predictions are

Figure 3.4: Sample images from the DLR-ACD dataset. The images cover large spatial areas and show different crowd densities within one single image.

Metric	Description
IDF1	ID F1-Score
IDP	ID Global Min-Cost Precision
IDR	ID Global Min-Cost Recall
Rcll	Recall
Prcn	Precision
FAR	False Acceptance Rate
GT	Number of Objects in Sequence
MT	Ratio of Mostly Tracked Objects
PT	Ratio of Partially Tracked Objects
ML	Ratio of Mostly Lost Objects
FP	False Positives
FN	False Negatives
ID	Number of Identity Switches
FM	Number of Fragmented Tracks
MOTA	Multiple Object Tracker Accuracy
MOTP	Multiple Object Tracker Precision
MOTAL	Multiple Object Tracker Accuracy Log

Table 3.4: Description of the metrics used for quantitative evaluation.

true positives (TP), describing an annotated object, or whether they are false positives (FP), missing the correct annotation. An annotated object that is missed by any prediction is a false negative (FN). In our case, predictions or so-called tracklets and annotations are associated as TPs if their intersection over union (IoU) score of their respective bounding boxes is greater than 0.5. An identity switch (ID) occurs if an annotated object a is associated with a tracklet t, and the previous assignment was $a \neq t$. The fragmentation metric shows the total number of times a trajectory is interrupted during tracking. For all of these measures, excluding TPs, applies that a higher number stands for a worse performance of the tracker.

However, the most crucial evaluation metrics are the multiple object tracker accuracy (MOTA) and the multiple object tracker precision (MOTP). MOTP gives a detailed account of a tracker's performance in estimating precise object locations.

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{3.1}$$

where $d_{t,i}$ is the distance between a matched object i to the ground truth annotation in frame t, and c is the total number of matched objects. MOTA describes a tracker's ability of keeping trajectories independant of the precision of the predictions.

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + ID_t)}{\sum_t GT_t} \tag{3.2}$$

The multiple object tracker accuracy log (MOTAL) is similar to MOTA, however, ID switches

are considered on a logarithmic scale.

$$MOTAL = 1 - \frac{\sum FN_T + FP_t + log_{10}(ID_t + 1)}{\sum GT_t} \tag{3.3}$$

Each tracklet can be assigned to mostly tracked (MT), partially tracked (PT), or mostly lost (ML). The decision is based on how successful an object is tracked during its whole lifetime. A tracklet is mostly lost if it is only tracked less than 20 % of its lifetime and mostly tracked if it is recovered more than 80 % of its lifetime. Partially tracked applies to all remaining tracklets. We give MT, PT, and ML as percentages from the total amount of tracks. The false acceptance rate (FAR) describes the average amount of FPs per frame.

$$FAR = \frac{\sum FP_t}{f} \tag{3.4}$$

,where f is the total number of frames.

Additionally, we also provide recall and precision measures. We define them as followed:

$$Rcll = \frac{\sum TP_t}{\sum(TP_t + FN_t)} \tag{3.5}$$

$$Prcn = \frac{\sum TP_t}{\sum(TP_t + FP_t)} \tag{3.6}$$

Identification precision (IDP), identification recall (IDR), and IDF1 scores are related. However, in contrast to recall or precision, IDP, IDR, and IDF1 scores take into account how long the tracker correctly identified the targets. The definition of IDP and IDR is the ratio of computed and ground-truth detections, respectively, that are correctly identified. The IDF1 is calculated as the ratio of correctly identified detections over the average number of computed and ground-truth detections, granting the possibility to rank a tracker based on a single scalar value. For any further information on these metrics, we refer to the work of Ristani et al. [89].

4 Preliminary Experiments

This chapter deals with different experiments we have taken in order to understand the difficulties of our aerial tracking scenarios. We provide insights into existing tracking methods combined with our data, show certain advantages, disadvantages, and failure cases. Finally, we conclude this chapter with common problems of existing trackers faced with aerial object tracking and present some ideas on how to solve them.

In the early phase of this thesis, only the KIT AIS pedestrian dataset was available to us. Hence, this chapter focuses on experiments with this dataset. Nevertheless, the findings and conclusion are also valid for the KIT AIS vehicle and the AerialMPT dataset since they share similar characteristics.

High tracking scores are usually correlated with good detections, at least in tracking-by-detection frameworks. Since this chapter deals mainly with tracking and not with detection, we assumed perfect detections and used the ground truth data, unless indicated otherwise. For detection-free methods, initial bounding boxes were given to the tracker, which propagates them to the object positions in the next frames. As a consequence, for detection-based methods, the most substantial measure is the number of ID switches. For other methods, all metrics are essential.

In the following subsections, we describe the experiments conducted in the scope of this thesis and provide some experimental results of trackers including KCF, MOSSE, CSRT, Median Flow, SORT, DeepSORT, Euclidean Online Tracking, and Stacked DCFNet.

4.1 From Single- to Multi-Object Tracking

Many tracking frameworks exist that were initially designed to track single objects only. However, most of them can easily be extended to handle multiple objects. An essential function of MOTs is track management: The trackers have to be able to save and exploit multiple active tracks at the same time, to delete the tracks of leaving, and to initialize the tracks of entering objects. We developed a Python class that is responsible for these actions and extended the SOT frameworks by multi-object compatibility. We use this class for all SOTs utilized within the scope of this thesis. It unites memory management, including the assignment of unique track IDs and individual object position storage, with track aging, deleting, and initializing functions. These functions can be called from outside the class, and hence, it is accessible to detail the explicit criteria for track creations and deletions in the algorithm itself.

Table 4.1: Results of MOSSE on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	8.0	8.1	7.9	9.1	9.2	78.08	94	1.1	5.3	93.6	1015	1032	0	9	-80.4	96.9	-80.4
AA_Walking_02	17	6.6	6.4	6.7	8.0	7.6	151.76	188	1.6	10.1	88.3	2580	2458	2	20	-88.7	95.7	-88.6
Munich02	31	4.3	4.2	4.5	5.7	5.4	199.68	230	0.9	4.3	94.8	6190	5775	29	78	-95.8	61.9	-95.4
RaR_Snack_Zone_02	4	29.4	29.2	29.6	30.4	30.0	153.25	220	99.5	219	0	613	602	0	14	-40.5	94.9	-40.5
RaR_Snack_Zone_04	4	25.8	25.7	25.9	27.0	26.8	226.25	311	0.3	99.7	0	905	898	0	12	-46.6	97.5	-46.6
Total	69	9.1	8.9	9.3	10.5	10.0	163.81	1043	0.8	54.0	45.2	11303	10765	31	133	-85.8	86.7	-83.5

Table 4.2: Results of KCF on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	8.1	8.1	8.0	9.1	9.2	78.08	94	1.1	6.4	92.5	1015	1032	0	8	-80.4	97.3	-80.4
AA_Walking_02	17	6.5	6.3	6.7	7.8	7.3	154.88	188	1.6	10.6	87.8	2633	2463	3	14	-90.9	96.9	-90.8
Munich02	31	4.3	4.1	4.4	5.6	5.2	201.74	230	0.9	3.9	95.2	6254	5781	29	75	-97.0	62.2	-96.5
RaR_Snack_Zone_02	4	29.3	29.1	29.5	29.8	29.5	154.50	220	1.8	98.2	0.0	618	607	0	8	-41.6	95.1	-41.6
RaR_Snack_Zone_04	4	25.8	25.7	25.9	26.9	26.8	226.50	311	0.3	99.7	0.0	906	899	0	11	-46.7	97.9	-46.7
Total	69	9.0	8.8	9.3	10.3	9.8	165.56	1043	1.1	53.8	45.1	11426	10782	32	116	-84.9	87.2	-84.7

4.1.1 KCF, MOSSE, CSRT & Median Flow

OpenCV provides several built-in object tracking algorithms. We experiment with the KCF [73], MOSSE [7], CSRT [77], and Median Flow [76] tracker. Initially, these trackers are made for SOT, but they can be easily extended to multi-object scenarios within the OpenCV framework. Initial bounding box positions are given as ground truth to the trackers, which locate the objects in the next frame and return their respective bounding box positions. We delete objects if they leave the image region, and their trackage is $a > 3$. We report the results of these trackers in Table 4.1, Table 4.2, Table 4.3 and Table 4.4.

However, all of these trackers perform poorly with total MOTA scores varying between -85.8 and -55.9. KCF and MOSSE results are very similar in general, with most metrics differing very little. The use of HOG features and non-linear kernels utilized in KCF improves MOTA by 0.9 and MOTP by 0.5 points compared to MOSSE, resulting in a total MOTA score of -84.9 and a MOTP score of 87.2. The ratio of mostly tracked object hovers at 1 % for both algorithms.

CSRT, also based on a DCF, outperforms both prior methods significantly, reaching a total MOTA and MOTP of -55.9 and 78.4. The ratio of mostly tracked objects almost reaches 10 % and proves the effectiveness of channel and reliability scores. Median Flow achieves a comparable performance with total MOTA and MOTP scores of -63.8 and 77.7.

All algorithms perform significantly better on the RaR_Snack_Zone sequences compared to the other ones. However, based on a visual inspection, we argue that this improvement is an immediate result of the short sequence length consisting of four frames only. Additionally, we argue that the low performance of these methods directly correlates with the use of handcrafted features.

4.1.2 Stacked DCFNet

DCFNet [90] is also a SOT based on a DCF. However, the DCF is implemented as part of a DNN and uses deep features from image crops extracted by a light-weight CNN. Consequently, this network is a perfect choice to study whether deep features improve the

Table 4.3: Results of CSRT on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	12.9	13.2	12.5	15.1	15.9	69.54	94	1.1	30.9	68.0	904	964	10	29	-65.5	84.6	-64.7
AA_Walking_02	17	9.2	10.0	8.5	11	12.9	116.88	188	2.7	15.4	81.9	187	2378	12	41	-63.9	88.0	-63.5
Munich02	31	9.2	9.9	8.7	10.9	12.5	151.45	230	1.8	14.3	83.9	4696	5455	66	137	-66.8	61.2	-65.8
RaR_Snack_Zone_02	4	43.2	42.0	42.5	43.8	43.3	124.25	220	17.3	82.7	0.0	497	486	0	16	-13.6	87.9	-13.6
RaR_Snack_Zone_04	4	45.6	45.5	45.0	47.9	47.6	162.00	311	16.7	83.3	0.0	648	641	3	31	-5.0	85.2	-4.8
Total	69	16.0	16.9	15.2	17.5	19.4	126.55	1043	9.6	51.0	39.4	8732	9924	91	254	-55.9	78.4	-55.1

Table 4.4: Results of Median Flow on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	27.3	27.3	27.4	28.5	28.3	62.85	94	1.1	68.1	30.8	817	812	4	49	-43.9	74.9	-43.6
AA_Walking_02	17	10.0	9.9	10.0	11.1	11.0	141.06	188	1.6	21.3	77.1	2398	2374	8	16	-79.0	86.3	-78.7
Munich02	31	9.2	9.0	9.4	9.9	9.5	186.39	230	1.3	8.7	90.0	5778	5517	10	53	-84.6	64.7	-84.4
RaR_Snack_Zone_02	4	51.7	51.4	52.0	52.8	52.2	104.75	220	8.6	91.4	0.0	419	408	2	14	4.2	83.7	4.3
RaR_Snack_Zone_04	4	53.1	53.0	53.3	53.9	53.6	143.5	311	17.4	82.6	0.0	574	567	6	29	6.7	83.0	7.2
Total	69	18.5	18.3	18.8	19.5	19.0	144.72	1043	7.7	55.8	36.5	9986	9678	30	161	-63.8	77.7	-63.5

tracking performance compared to the handcrafted ones or not.

We took the *PyTorch* implementation[1] of DCFNet as a baseline and modified the network structure, so that the network is capable of multi-object tracking. We call the multi-object version of the network "Stacked DCFNet".

From the KIT AIS pedestrian training set, we create image crops for each pedestrian, with each individual centered in the middle of the crop. The crop size is dependant on the bounding box size, which is multiplied by a factor of 10 to obtain the final scene context. Afterwards, we rescale all crops to 125 x 125 pixels and save them. This process results in 20,666 image crops. We retrain the convolutional layers of the network with these crops and apply ADAM [91] optimizer with an initial learning rate of 0.01 and a mini-batch size of 64. We train the network for 50 epochs with the MSE loss and set the spatial bandwidth to 0.1 for both, online tracking and offline training. We use the previously mentioned *Python* class for track management, and similar to the OpenCV methods, we remove tracks that are leaving the image region when their age is $a > 3$. Multiple targets are given to the network within one batch. For each target, the network receives two image crops with both crops centered on the known previous position of the object. The network tracks the target by defining the network output as the probability heatmap of object location. The highest value represents the most likely object position in the current image crop. If this value is below a threshold t, we consider the object as lost.

Furthermore, in contrast to the original work, we propose a simple linear motion model and set the center point of the search crop to the position estimate of this model instead of the position of the target crop. Based on the last movement $v_t(x, y)$ of a target, we estimate its position as followed.

$$p_{est}(x, y) = p(x, y) + k \cdot v_t(x, y) \tag{4.1}$$

, where k is a hyperparameter that models the influence of the last movement.

We present the results of Stacked DCFNet in Table 4.5. The MOTA score improves significantly compared to the previously tested methods, increasing by 18.6 points to a final

[1] https://github.com/foolwood/DCFNet_pytorch

Table 4.5: Results of Stacked DCFNet on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	41.9	42.4	41.3	42.7	43.9	47.77	94	12.8	58.5	28.7	621	650	15	71	-13.3	74.7	-12.1
AA_Walking_02	17	31.4	31.6	31.2	32.3	32.7	104.29	188	5.9	45.7	48.4	1773	1809	23	184	-35.0	74.1	-34.2
Munich02	31	21.2	20.6	21.9	25.0	23.6	160.45	230	1.7	50.0	48.3	4974	4591	97	322	-57.7	60.5	-56.2
RaR_Snack_Zone_02	4	51.8	52.3	51.3	52.4	53.4	99.00	220	22.3	74.5	3.2	396	412	4	35	6.1	84	6.5
RaR_Snack_Zone_04	4	51.8	52.6	51.0	52.1	53.7	138.00	311	21.9	74.9	3.2	552	589	0	39	7.2	83.6	7.2
Total	69	30.0	30.2	30.9	33.1	32.3	120.52	1043	13.8	62.6	23.6	8316	8051	139	651	-37.3	71.6	-36.1

total MOTA of -37.3 compared to CSRT. Stacked DCFNet achieves a competitive MOTP score of 71.6. The ratio of mostly tracked and mostly lost tracks is also improving, only loosing 23.6 % of all tracks while tracking 13.8 % mostly. The tracker's performance on *AA_Crossing_02*, *AA_Walking_02* and *Munich02* increases the most compared to the remaining sequences. This gain is important since it shows the tracker's ability to keep tracks during longer periods in sequences with more frames. Compared to CSRT, all metrics except IDs and FMs show significant improvements.

All in all, the results indicate that deep features outperform handcrafted ones by a large margin.

4.2 Multi-Object Trackers

This section deals with the MOTs we experimented with during this thesis. MOTs include SORT, DeepSORT, and Tracktor++. Additionally, we implemented a new tracking algorithm called Euclidean Online Tracking (EOT) based on SORT. EOT uses the Euclidean distance for tracklet-detection matching.

4.2.1 SORT & DeepSORT

DeepSORT [60] is an MOT, that composes deep appearance features and IoU based tracking. We took the *PyTorch* implementation[2] of DeepSORT and adapted it to work with our dataset. We based our experiments on the ground truth and did not use the object detector that is part of DeepSORT.

In the first experiment, we take DeepSORT and run it on the KIT AIS pedestrian dataset without any changes. The results in Table 4.6 show that the tracker with standard parameter settings is not suitable to track small objects in aerial imagery. The tracker uses appearance features to associate objects to tracklets; however, for the first l frames, objects are associated using an IoU metric until enough appearance features are available. Hence, DeepSORT requires objects to be successfully tracked by SORT for the first frames. The regular value for the IoU threshold is 0.5. The standard DeepSORT uses a Kalman filter for each object to estimate its position in the next frame. However, there is little IoU overlap between most predictions and detections, and many tracks can not be associated with any detection, making it also impossible to use the deep features afterwards. The tiny bounding box size especially causes the little overlap. For example, if the Kalman filter estimate is shifted more than 2

[2]https://github.com/ZQPei/deep_sort_pytorch

Table 4.6: Results of DeepSORT with default settings on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	3.1	3.1	3.1	100.0	100.0	0.00	94	100.0	0.0	0.0	0	0	940	1	17.2	99.7	99.7
AA_Walking_02	17	7.7	7.7	7.8	100.0	98.9	1.71	188	100.0	0.0	0.0	29	0	2145	5	18.6	99.0	98.8
Munich02	31	9.1	8.8	9.4	100.0	92.8	15.42	230	100.0	0.0	0.0	478	0	4681	1	15.8	64.0	92.1
RaR_Snack_Zone_02	4	21.0	20.9	21.2	100.0	98.7	2.75	220	100.0	0.0	0.0	11	0	351	2	58.2	98.1	98.4
RaR_Snack_Zone_04	4	17.9	17.9	18.0	100.0	99.6	1.25	311	100.0	0.0	0.0	5	0	510	0	58.1	98.6	99.4
Total	69	10.0	9.8	10.2	100.0	95.8	7.58	1043	100.0	0.0	0.0	523	0	8627	9	23.9	81.1	98.6

pixels from the ground truth position, for bounding boxes with a size of 4×4 pixels, there is not enough overlap to consider tracklet and object as a match. This mismatch results in high falsely initiated new tracks, leading to a total amount of 8,627 ID switches, an average amount of 8.271 ID switches per person, and an average amount of 0.717 ID switches per detection. Rcll, Prcn, FAR, MT, PT, ML, FN, MOTP, and FM have low significance since the experiments are based on the ground truth. This assumption is valid for the rest of the DeepSORT and SORT experiments.

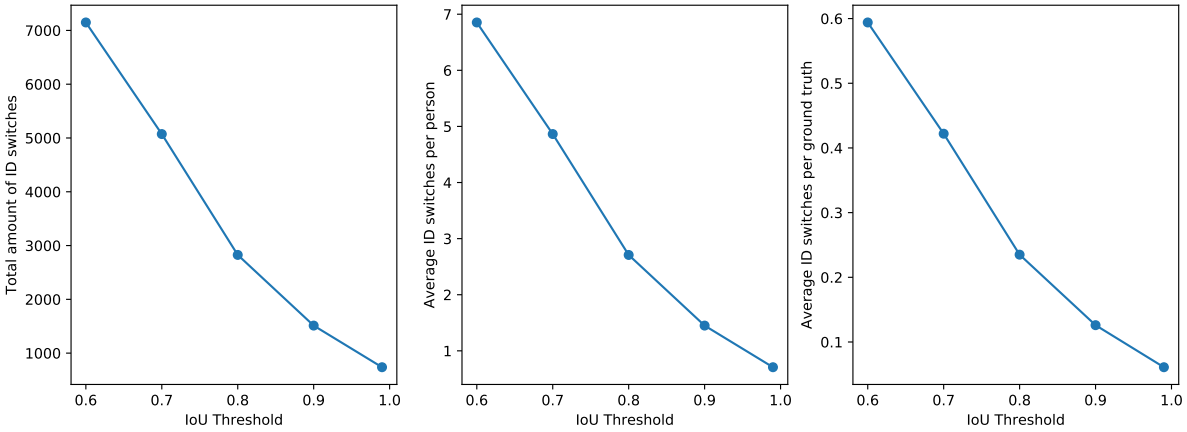


Figure 4.1: Correlation between ID switches and IoU threshold with DeepSORT.

We tackle this problem by enlarging the given ground truth bounding boxes by factor two. We assume this will increase the IoU overlap, resulting in the confirmation of tracklets and using of appearance information. The results of this experiment are visualized in table Table 4.7. The total amount of ID switches decreased from 8627 to 5073, which is a 41.196 % decrease. The average amount of ID switches per person decreased to 4.864, the average amount of ID switches per detection to 0.422. In Figure 4.1, we visualized the number of ID switches with different IoU thresholds. It is visible that increasing the threshold, thereby minimizing the required overlap for detections and predictions to be associated as matches, reduces the number of ID switches. Setting the IoU threshold to 0.99 reduces the ID switches the most, resulting in a total amount of 738 switches. We visualize the results in Table 4.8. FP rate decreases to 502, FN increased to 75. Both changes are a result of the increased track confirmations. These experiments confirm our assumptions that the missing IoU overlap is

Table 4.7: Results of DeepSORT with default settings and doubled bounding box sizes on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	34.8	34.5	35.1	100.0	98.4	1.38	94	100.0	0.0	0.0	18	0	566	1	48.5	94.3	98.2
AA_Walking_02	17	46.6	46.0	47.1	100.0	98.8	3.59	188	100.0	0.0	0.0	61	0	1073	5	57.5	93.1	97.6
Munich02	31	29.5	27.6	31.5	100.0	87.7	27.71	230	100.0	0.0	0.0	859	0	2989	1	37.2	63.9	85.9
RaR_Snack_Zone_02	4	52.2	51.9	52.5	100.0	98.9	2.5	220	100.0	0.0	0.0	10	0	203	2	75.4	95.7	98.6
RaR_Snack_Zone_04	4	61.2	61.0	61.5	100.0	99.2	2.5	311	100.0	0.0	0.0	10	0	242	0	79.5	94.4	99.0
Total	69	38.4	36.9	39.9	100.0	92.6	13.88	1043	100.0	0.0	0.0	958	0	5073	9	49.9	78.7	92.0

Table 4.8: Results of DeepSORT with IoU threshold set to 0.99 and doubled bounding box sizes on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	93.8	92.5	95.2	99.8	96.9	2.77	94	100.0	0.0	0.0	36	2	45	2	93.8	85.0	96.5
AA_Walking_02	17	88.7	84.4	93.4	99.7	90.0	17.35	188	100.0	0.0	0.0	295	8	42	12	87.0	86.4	88.6
Munich02	31	73.1	70.9	75.3	98.9	93.2	14.23	230	100.0	0.0	0.0	441	67	565	56	82.5	62.9	91.7
RaR_Snack_Zone_02	4	90.1	89.9	90.4	99.8	99.2	1.75	220	99.1	0.9	0.0	7	2	37	4	94.7	87.9	98.8
RaR_Snack_Zone_04	4	90.2	90.1	90.3	100.0	99.8	0.75	311	100.0	0.0	0.0	3	0	49	0	95.8	88.4	99.6
Total	69	82.1	80.7	83.6	99.4	96.0	7.28	1043	1041	99.8	0.2	502	75	738	70	89.1	74.7	95.2

the main issue of the standard DeepSORT tracker. To further confirm this finding, we also test this setting on the scenario with original bounding box sizes, also achieving an ID switch decrease by 53.530 % to 4,009 and a MOTA increase by 31.5 points to 55.4.

With the improved tracklet-to-object association, appearance features should play a more prominent role after tracklets were successfully tracked for the first l frames. Additionally, since appearance-based matching is the priority in DeepSORT, an important step is fine-tuning the featuring extracting neural network with our data. Originally, the network was trained at a large person re-identification dataset. Our scenario differs a lot, as persons are only visible from an aerial view, and the resolution of each person in the pictures is much lower than the resolution available in the re-identification dataset. Also, bounding boxes are much smaller compared to the original task, where each bounding box was scaled to 128×64 pixels, representing a common pedestrian form. Upscaling from 4×4 pixels, the average bounding box size in our dataset, to 128×64 pixels leads to heavy interpolation errors. Hence, we decided to finetune the network with the training set of the KIT AIS pedestrian dataset. The last re-identification layers were initialized newly, the rest of the network with the provided weights and biases. We also changed the number of classes to 610, which represents the number of different persons after successfully cropping the images to the bounding box size and ignoring crops too close to the edge of the images. Instead of upscaling to 128×64 pixels, we only scale up to 50×50. We trained the classifier with the provided train class, used SGD

Table 4.9: Results of DeepSORT with IoU threshold set to 0.99 and regular bounding box sizes on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	55.0	54.4	55.6	99.0	96.9	2.77	94	100.0	0.0	0.0	36	11	347	10	65.3	83.6	95.6
AA_Walking_02	17	63.4	62.5	64.3	99.1	96.3	6.06	188	100.0	0.0	0.0	103	23	557	25	74.4	82.0	95.2
Munich02	31	24.2	22.8	25.8	97.2	85.8	31.77	230	99.6	0.4	0.0	985	170	2737	151	36.5	62.9	81.1
RaR_Snack_Zone_02	4	57.7	57.3	58.2	100.0	98.5	3.25	220	100.0	0.0	0.0	13	0	177	2	78.0	90.4	98.2
RaR_Snack_Zone_04	4	69.1	68.7	69.5	99.9	98.8	3.75	311	99.7	0.3	0.0	15	1	191	1	83.2	87.2	98.5
Total	69	43.3	40.8	44.0	98.3	91.1	16.7	1043	99.8	0.2	0.0	1152	205	4009	189	55.4	73.7	88.7

Table 4.10: Results of DeepSORT with IoU threshold set to 0.99, doubled bounding box sizes and finetuned network on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	93.1	92.7	93.4	100.0	99.3	0.62	94	100.0	0.0	0.0	8	0	43	1	95.5	85.1	99.2
AA_Walking_02	17	93.1	92.4	93.7	99.8	98.4	2.53	188	100.0	0.0	0.0	43	6	42	9	96.6	86.5	98.1
Munich02	31	73.3	71.2	75.5	99.0	93.3	13.94	230	100.0	0.0	0.0	432	63	563	54	82.7	62.9	91.9
RaR_Snack_Zone_02	4	90.1	89.9	90.4	99.8	99.2	1.75	220	99.1	0.9	0.0	7	2	37	4	94.7	87.9	98.8
RaR_Snack_Zone_04	4	90.2	90.1	90.3	100.0	99.8	0.75	311	100.0	0.0	0.0	3	0	49	0	95.8	88.4	99.6
Total	69	82.4	81.0	83.8	99.4	96.0	7.14	1043	99.8	0.2	0.0	493	71	734	68	89.2	74.7	95.3

Table 4.11: Results of SORT with IoU threshold set to 0.99 and doubled bounding box sizes on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	93.1	92.7	93.4	100.0	99.3	0.62	94	100.0	0.0	0.0	8	0	45	1	95.3	85.0	99.1
AA_Walking_02	17	94.5	93.9	95.1	99.3	98.6	2.18	188	100.0	0.0	0.0	37	2	30	6	97.4	86.5	98.5
Munich02	31	80.4	79.6	81.3	99.3	97.2	5.68	230	100.0	0.0	0.0	176	42	284	37	91.8	63.0	96.4
RaR_Snack_Zone_02	4	90.5	90.2	90.8	99.8	99.2	1.75	220	99.1	0.9	0.0	7	2	34	4	95.0	87.9	98.8
RaR_Snack_Zone_04	4	90.5	90.4	90.7	100.0	99.8	0.75	311	100.0	0.0	0.0	3	0	45	0	96.1	88.4	99.6
Total	69	86.5	85.5	87.2	99.6	98.1	3.35	1043	99.8	0.2	0.0	231	46	438	48	94.1	74.7	97.7

with a mini-batch size of 128, set the initial learning rate to 0.01, and trained for 20 epochs with Cross-Entropy-Loss. We use doubled bounding box sizes during this experiment.

In Table 4.10, we present the results of this experiment. The total amount of ID switches decreases from 738 to 734 only. We argue that the method, which models appearance features in DeepSORT, is not suitable for our case. For large objects, small deviations of the bounding box positions are tolerable since most parts of the bounding box are still object-relevant. In our scenario, small deviations can cause significant changes in object relevance. The extracted features do not necessarily correspond to the object, but in large parts to the background. Consequently, in the appearance matching step, historic features and current features can differ significantly, causing high distances when comparing these for target association. Additionally, the appearance features of different persons are often not discriminative enough to distinguish between them.

To confirm this theory, we also test DeepSORT without any appearance information, also known as SORT. In Table 4.12 and Table 4.11 we provide the results with normal bounding box sizes and doubled bounding box sizes using an IoU threshold of 0.99. In this case, SORT outperforms the finetuned DeepSORT, reaching 3805 and 438 ID switches, respectively. The amount of FP drop to 840 and 231, the number of FNs to 151 and 46. Nevertheless, the amount of ID switches is still high, given that we use the ground truth positions.

We conclude that the low frame rate and the sizes of the small objects make IoU an uninformed metric, not fitting to our scenarios. Enlarging the bounding boxes was essential to study the impact of IoU matching and object size. Nevertheless, enlarging the bounding boxes sizes can not be a final solution, leading to a decreased localization accuracy since objects can be placed anywhere in the enlarged bounding box. Hence, in the following, we will only use regular bounding box sizes.

Table 4.12: Results of SORT with IoU threshold set to 0.99 and regular bounding box sizes on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	55.9	55.4	56.5	99.1	97.2	5.54	94	100.0	0.0	0.0	33	10	343	9	66.0	83.5	96.0
AA_Walking_02	17	64.0	63.2	64.9	99.3	96.7	5.29	188	100.0	0.0	0.0	90	19	550	21	75.3	82.0	95.8
Munich02	31	24.6	23.6	25.8	98.0	89.7	22.23	230	99.6	0.4	0.0	689	122	2544	108	45.2	62.8	86.7
RaR_Snack_Zone_02	4	57.7	57.3	58.2	100.0	98.5	3.25	220	100.0	0.0	0.0	13	0	177	2	78.0	90.4	98.2
RaR_Snack_Zone_04	4	69.1	68.7	69.5	99.9	98.8	3.75	311	99.7	0.3	0.0	15	1	191	1	83.2	87.2	98.5
Total	69	42.9	41.8	44.2	98.7	93.4	12.17	1043	99.8	0.2	0.0	840	151	3805	141	60.1	73.6	91.7

Table 4.13: Results of EOT with Euclidean distance set to 17 pixel and regular bounding box bizes on KIT AIS edestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	94.4	94.4	94.4	95.3	95.2	4.15	94	91.5	8.5	0.0	54	53	4	34	90.2	73.8	90.5
AA_Walking_02	17	94.6	94.0	95.1	96.9	95.8	6.71	188	96.8	2,7	0.5	114	82	10	63	92.3	76.6	92.6
Munich02	31	76.0	75.8	76.2	77.0	76.5	46.65	230	44.3	54.8	0.9	1446	1409	15	930	53.1	60.4	53.4
RaR_Snack_Zone_02	4	95.0	94.9	95.1	96.5	96.3	8.00	220	87.7	12.3	0.0	32	30	3	16	92.5	77.6	92.8
RaR_Snack_Zone_04	4	95.2	95.1	95.2	96.3	96.3	11.50	311	76.2	23.8	0.0	46	45	5	31	92.2	78.6	92.5
Total	69	85.2	84.9	85.5	86.5	86.0	24.52	1043	80.2	19.6	0.2	1692	1619	37	1074	72.2	69.3	72.5

4.2.2 Euclidean Online Tracking

We identify two main problems with DeepSORT and SORT paired with our scenario: Appearance association is used as a priority, and missing IoU overlap leads to unconfirmed tracklets. Nevertheless, the simplicity of DeepSORT and SORT is also a key advantage.

We decide to adopt the SORT architecture to our needs. We leave the Kalman filter untouched and use it for motion prediction similar to in SORT. Instead of using appearance information as first matching priority, we decide to calculate the euclidean distance between our predictions and the detections (i.e., the ground truth) and base our associations on this measure. We calculate the distance between all predictions, and all detections normalized w.r.t. the GSD and construct a cost matrix.

$$D_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \cdot GSD \tag{4.2}$$

Similar to SORT, we use the Hungarian algorithm to solve for the minimal global cost. However, considering the distance metric, the Hungarian algorithm can lead to an error propagation throughout all tracklet-object associations in cases where objects leave or enter the scene. Hence, we restrict the cost matrix: All distance greater than a threshold u are ignored and set to an infinity cost. The threshold u is found experimentally and set to $u = 17 \cdot GSD$. This method solves the error propagation problem successfully. Furthermore, only tracks that were successfully tracked in the previous frame are allowed to be associated with this method since the uncertainty of the Kalman filter increase when it is not updated with a measurement. We refer to this method as EOT in the following.

We visualized the results in Table 4.13. The total MOTA score increases from 60.1 to 72.2 points, while IDs decline significantly from 3805 to 37. FPs and FNs increase to 1692 and 1619, respectively; however, the main reason for this behavior is the better keeping of tracks leading to small deviations in the Kalman filter predictions. Furthermore, the results indicate that the euclidean distance is a suitable associating measure in our scenario.

4.2.3 Tracktor++

To explore another tracker that is building upon deep features, we also evaluate our dataset with Tracktor++ [2]. We use the *PyTorch* implementation[3] and adapt it to our needs. As explained in subsection 2.3.1, Tracktor++ uses a Faster RCNN to perform both, object detection and tracking via regression.

We evaluate the tracker without any changes on the KIT AIS pedestrian dataset; however, instead of using the object detector as a detection module, we give object position as ground truth positions. Tracktor++ is not able to detect any kind of tracks, and evaluation fails with an error. The Faster RCNN was originally not trained to our kind of data; thus, the vast amount of objects, the small size of the objects, and the different recording and general data scenario requires retraining and finetuning. This process is necessary since not only the detection part but also the tracking part depends on the regression head of the network. The authors provide training utilities; nevertheless, we change them massively. We use Faster RCNN with a ResNet50 backbone pretrained on the ImageNet dataset. We change the anchor sizes to 2,3,4,5, and 6 and the aspect ratios to 0.7, 1.0, and 1.3, enabling the detection of small objects. Additionally, we increase the maximum detections per image to 300, set the minimum size of an image to be rescaled to 400 pixels, the region proposal non-maximum suppression (NMS) threshold to 0.3, and the box predictor NMS threshold to 0.1. The NMS thresholds influence the amount of overlap for region proposals and box predictions. Instead of SGD, we decide to use ADAM with an initial learning rate of 0.0001 and a weight decay of 0.0005. Additionally, we decrease the learning rate every 40 epochs by multiplying 0.1 and set the number of classes to two, representing background and pedestrians. We apply substantial online data augmentation, and flip the images with a probability of 0.5 respectively horizontally and vertically, apply color jitter and random scaling in a range of 10 %.

In Table 4.14, we show the results with Tracktor++ and the finetuned Faster RCNN. For object detection, Faster FCNN achieves a recall of 31 % and a precision of 25 %. The bad object detection performance is transferred into the tracking part. Tracktor++ only reaches a total MOTA of 5.3 with object detections based on the ground truth. This result is a decline of 67 points compared to EOT, which reaches a total MOTA of 72.2. The number of ID switches increases to 2188 compared with 37 when using EOT.

Tracktor++ has difficulties in dealing with low framerates and small objects. Targets move too much between frames, and the regression head is not able to regress the new position accurately.

4.2.4 Conclusion of Experiments

In the last subsections, we evaluated different tracking approaches and their applicability to our scenario. Table 4.15 compares the tracking results for all trackers.

For all algorithms based on tracking-by-detection, we used the ground truth as given detections. We initiated the first object positions based on the ground truth and left it to the tracker to regress the next positions for every other approach. In the following,

[3] https://github.com/phil-bergmann/tracking_wo_bnw

Table 4.14: Results of Tracktor++ and finetuned Faster RCNN on the KIT AIS pedestrian dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	12.7	19.6	9.4	48.2	100.0	x	94	20.1	51.1	28.8	0	588	432	107	10.1	0.13	–
AA_Walking_02	17	10.7	27.5	6.7	23.2	95.8	x	188	3.2	43.1	53.7	27	2050	426	154	6.3	0.13	–
Munich02	31	7.8	16.7	5.1	22.7	74.5	x	230	2.2	41.3	56.6	746	4736	965	412	-0.8	0.078	–
RaR_Snack_Zone_02	4	33.8	54.5	24.5	40.2	89.5	x	220	17.7	45.5	36.8	41	517	134	27	20.	0.091	–
RaR_Snack_Zone_04	4	32.5	50.2	24.0	42.9	89.8	x	311	22.2	44.1	33.7	60	702	231	25	19.3	0.064	–
Total	69	13.7	27.3	9.2	28.5	85.0	x	1043	13.2	44.2	42.6	604	8593	2188	725	5.3	0.095	–

Table 4.15: Performance comparison of different trackers and regular bounding box bizes

Method	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
SMSOT-CNN (*Caffe*)	34.0	33.2	34.9	38.2	36.4	116.40	1043	25.0	52.5	22.5	8028	7427	157	614	-29.8	71.0	-28.5
Mosse	9.1	8.9	9.3	10.5	10.0	163.81	1043	0.8	54.0	45.2	11303	10765	31	133	-85.8	86.7	-83.5
KCF	9.0	8.8	9.3	10.3	9.8	165.56	1043	1.1	53.8	45.1	11426	10782	32	116	-84.9	87.2	-84.7
CSRT	16.0	16.9	15.2	17.5	19.4	126.55	1043	9.6	51.0	39.4	8732	9924	91	254	-55.9	78.4	-55.1
Medianflow	18.5	18.3	18.8	19.5	19.0	144.72	1043	7.7	55.8	36.5	9986	9678	30	161	-63.8	77.7	-63.5
SORT	42.9	41.8	44.2	98.7	93.4	12.17	1043	99.8	0.2	0.0	840	151	3805	141	60.1	73.6	91.7
DeepSORT	43.3	40.8	44.0	98.3	91.1	16.7	1043	99.8	0.2	0.0	1152	205	4009	189	55.4	73.7	88.7
EOT	85.2	84.9	85.5	86.5	86.0	24.52	1043	80.2	19.6	0.2	1692	1619	37	1074	72.2	69.3	72.5
Tracktor++	13.7	27.3	9.2	28.5	85.0	8.75	1043	13.2	44.2	42.6	604	8593	2188	725	5.3	0.095	–
Stacked DCFNet	30.0	30.2	30.9	33.1	32.3	120.52	1043	13.8	62.6	23.6	8316	8051	139	651	-37.3	71.6	-36.1

this subsection evaluates the most exciting trackers more intensely and shows some key advantages and disadvantages. Afterwards, we give a conclusion to find the most promising research direction.

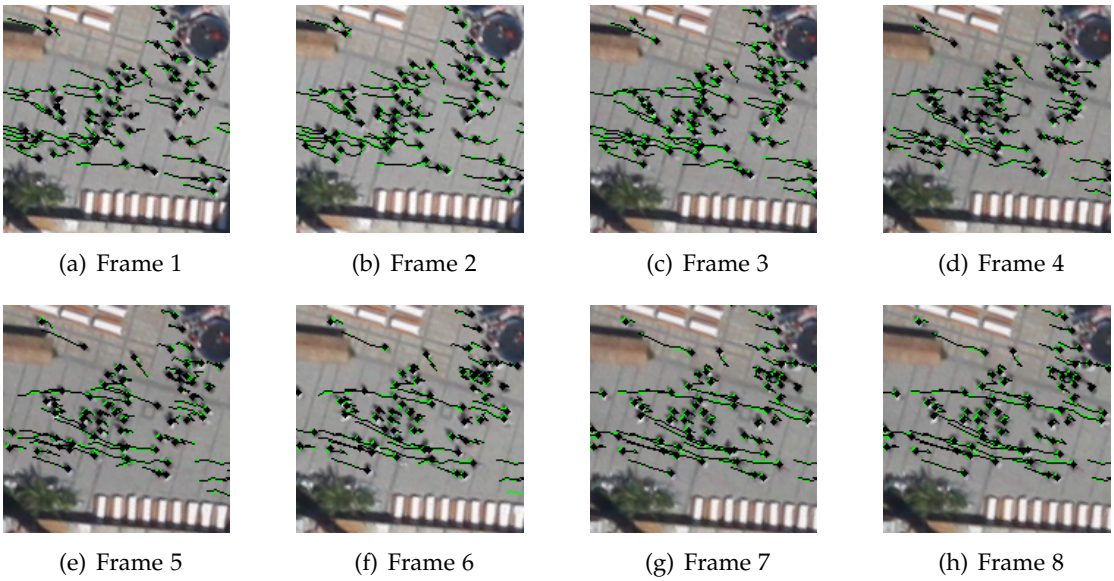
EOT showed the best tracking results in our experiments; hence we start with this tracker.

We visualize the major cases of successful tracking with EOT and detections based on ground truth and regular bounding box sizes in Figure 4.2. Tracklets are visualized in green color, the ground truth positions in black color. We make two important observations: Almost all objects are tracked successfully, even though the sequence is crowded and people walk in different directions. Furthermore, the significant cases of false positives and negatives are "wrong", e.g., EOT tracks most objects, but the overlap is less than half of the bounding box size, which is 4 square pixels.

In Figure 4.3, we visualized a typical failure case of the stacked DCFNet architecture. The tracker tracks most objects correctly. Nevertheless, an error is shown in the middle of the image crop, starting in frame 5. The tracker mistakes the line on the ground with the people walking across this line and loses both objects. We assume that the line shares similar appearance features with the tracked people.

In Figure 4.4, we show another typical failure case of DCFNet. The image crops show many people close to each other walking in different directions, a difficult task for the tracker. The vast amount of people leads to many ID switches in the consecutive frames. Similar to the first failure case, we assume that many people share the same appearance features.

We confirm both findings by visualizing the activation map of the last convolution layer of the network in Figure 4.5. The convolutional layers of DCFNets architecture were trained on people in our training dataset only, but the activation map also highlights the line on the ground in a similar way. The similar activation leads the tracker to mistake background objects or other persons with a given tracklet. Furthermore, based on our visualizations, each convolutional channel shows a different strength of activation. Based on this behavior, we



(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

(e) Frame 5 (f) Frame 6 (g) Frame 7 (h) Frame 8

Figure 4.2: Success case of EOT, test sequence *Munich02*.

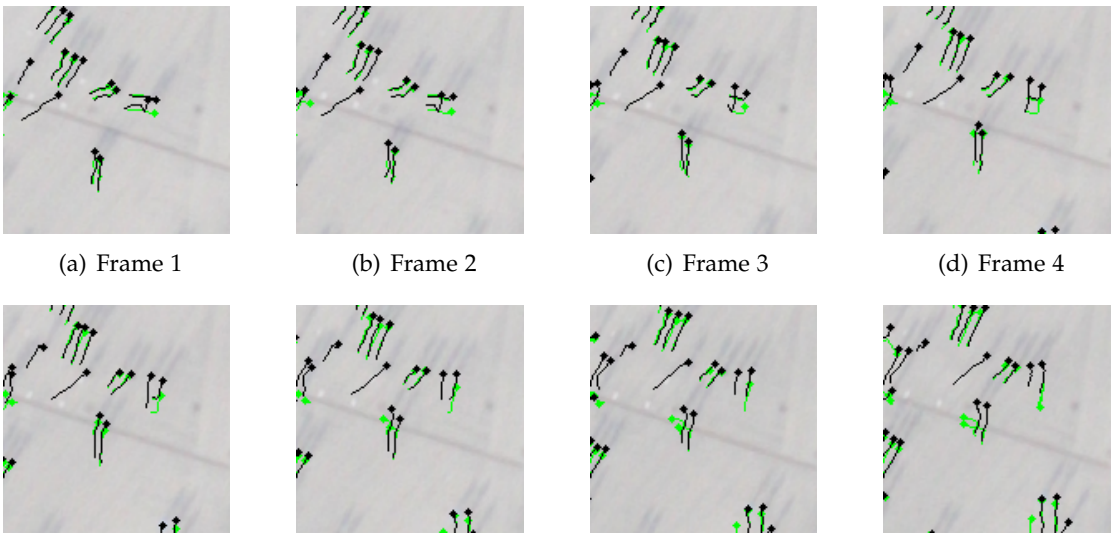
argue that each channel has a different importance with respect to the final output.

In Figure 4.6 we show a successful tracking case. People are not walking together as crowded as before, and the background is more distinguishable from the objects.

We also evaluate the SMSOT-CNN architecture by Bahmanyar et al. [30] and find that it shares the same problems with the stacked DCFNet, although achieving a slightly better total MOTA of -29.8 compared to -30.9. The similarity is reasonable since it also uses convolution layers to extract appearance information. Based on a visual inspection, SMSOT-CNN shares similar failure and success cases; however, it provides slightly better results.

In general, the euclidean distance paired with trajectory information is a more informed metric than IoU in our scenarios. Nevertheless, the quality of detections is significantly worse compared with the ground truth when using object detectors, as shown in subsection 4.2.3. This makes a model based on object detections and distance matching not suitable for our scenarios. Approaches that focus on deep appearance features used for re-identification share similar problems with object detectors, and indiscriminative features and similar appearance information lead to ID switches and loosing of tracks. Regression and correlation trackers, in our case the stacked DCFNet and SMSOT-CNN architecture, show in general a better performance than methods based on re-identification since they work with local crops where errors can not spread through the whole image. Nevertheless, they perform badly when confronted with similar-looking objects, crowded scenes, or occlusions. In general, we find that the route taken by a pedestrian is influenced by three main points: By its route history, by the movement and positions of surrounding people, and by the arrangement of the scene.

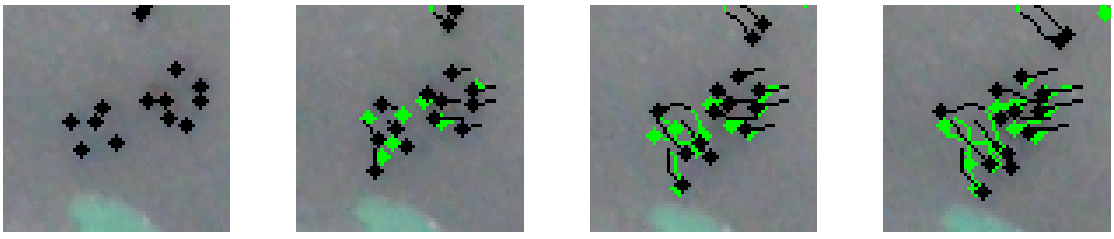
After carefully considering all of these aspects, we conclude that regression and correlation-based trackers can be improved by taking into account trajectory information and the influence of surrounding people. Hereby, typical failure cases shown in Figure 4.3 and Figure 4.4 can



(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

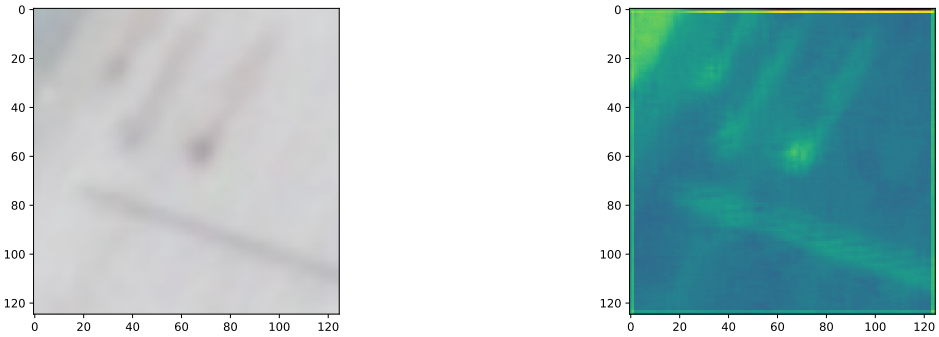
(e) Frame 5 (f) Frame 6 (g) Frame 7 (h) Frame 8

Figure 4.3: Failure case of stacked DCFNet, test sequence *AA_Walking_02*. Tracking results are visualized in green color, ground truth in black color.



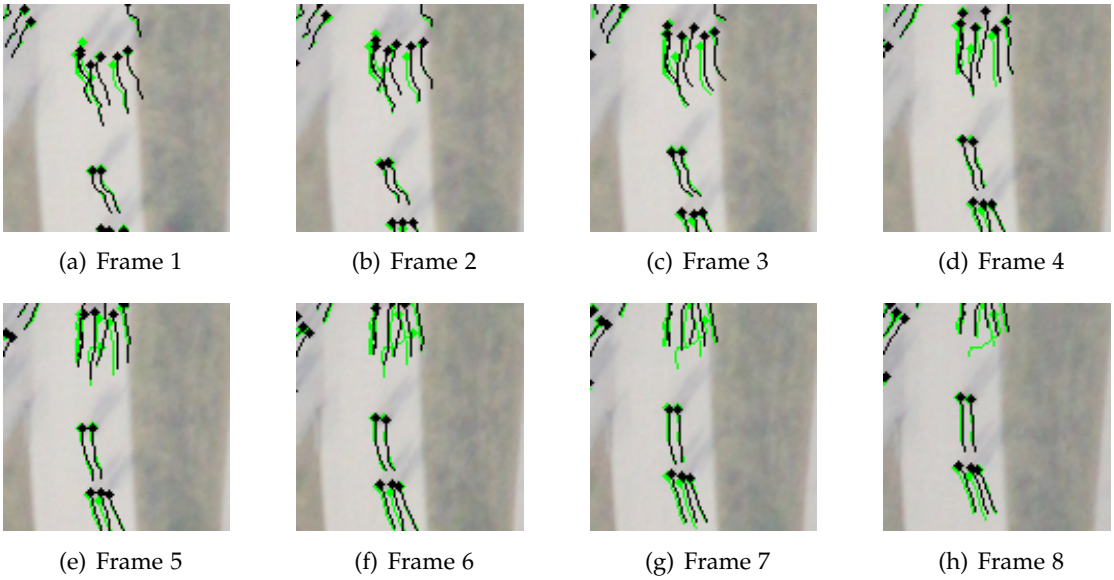
(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

Figure 4.4: Failure case of Stacked DCFNet, test sequence *RaR_Snack_Zone_04*



(a) Input (b) Activation Map

Figure 4.5: Activation map of the last conv layer of Stacked DCFNet



(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

(e) Frame 5 (f) Frame 6 (g) Frame 7 (h) Frame 8

Figure 4.6: Success case of Stacked DCFNet, test sequence *AA_Crossing_02*

be eliminated.

5 Methodology

This chapter summarizes our methodology and explains the proposed DL-based architecture. We propose two novel strategies to handle motion characteristics and adjacent neighbor modeling in an end-to-end trainable fashion. To conclude this chapter, we present our experiment and training settings.

As stated previously, scene arrangement, adjacent people, and an object's previous movement influence a pedestrian's trajectory mainly. The same occurs in many traffic scenarios, such as car or street traffic, which is even reinforced by the fact that vehicles move along predetermined paths (e.g., streets, highways, rails), unlikely to leave these. However, different objects have different motion characteristics: One may move slowly but steady, another one fast but discontinuously. For example, several studies have proven that walking speed of pedestrians is strongly influenced by different factors such as age, gender, temporal variations, cell phone usage, movement in groups, and even city size [92, 93]. In street traffic, similar aspects can influence driving behavior and lead to different movement characteristics, including phone usage, age, stress, and fatigue [94, 95]. Furthermore, vehicles directly influence other vehicles with their actions (e.g., if the Vehicle in front brakes, the following vehicles must brake, too).

The understanding of individual motion patterns is crucial for well-performing tracking algorithms, especially when only limited visual information about target objects is available. However, current regression-based tracking methods such as GOTURN and SMSOT-CNN do not incorporate the movement history or relationship between adjacent objects. These networks achieve tracking by monitoring a specific neighborhood of the target object, and hence, the network is not provided with the contextual information from outside the neighborhood. Additionally, the networks do not learn to distinguish between target objects and objects sharing similar appearance features within the crop. Identity switches and lost tracks are a consequence in crowded situations or object crossing, as we have shown in chapter 4. Other frameworks, such as DeepSORT or Tracktor++, employ motion models or Kalman filters to encode the previous movement. Nevertheless, such motion models are outdated. Therefore, we replace them with DL-based structures.

Tackling the limitations of previous work, we propose to fuse track history, the relationships of adjacent objects, and visual features in an end-to-end fashion within a regression-based DNN. Our approach is called AerialMPTNet and takes two regional crops of two consecutive images (previous and current), called target and search crop in which the object location is known and has to be detected, respectively. The size of both crops, and consequently, the amount of context the network receives, is dependent on the object's size. Both crops are centered on the known object coordinates of the previous frames and scaled to 227 x 227 pixels afterwards. Both crops are given to an SNN module, consisting of two branches

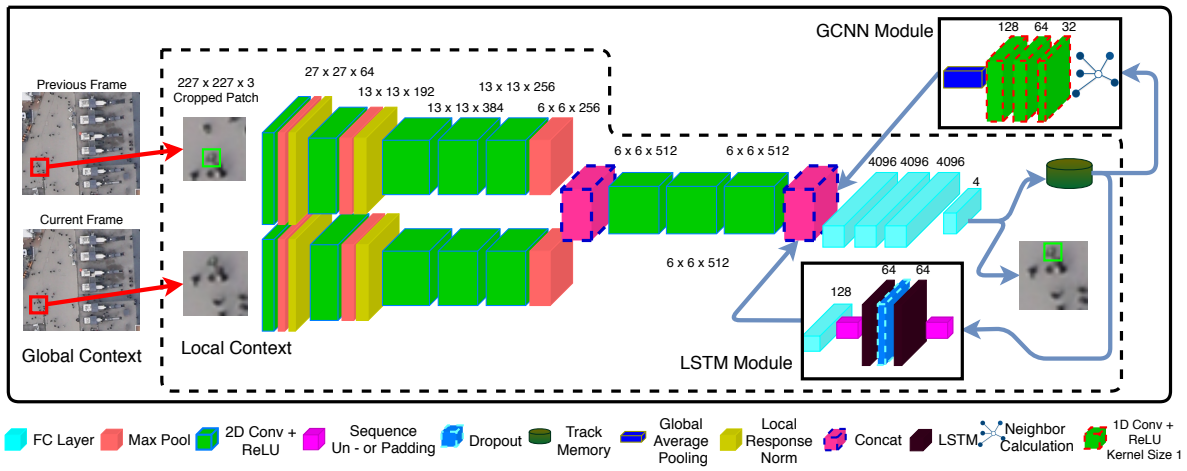


Figure 5.1: Overview of the network's architecture composing a SNN, a LSTM and a GraphCNN module. The inputs are two consecutive images cropped and centered to a target object, and the output is the object location in search crop coordinates.

sharing layer weights. The SNN module consists of five 2D convolutions, two local response normalization, and three max-pooling layers. It serves as a baseline, retained from the work of Bahmanyar et al. [30]. The output features Out_{SNN} are concatenated and given to four fully connected layers regressing the object position in search crop coordinates. The output of the network is four numbers representing the top-left and bottom-right coordinates of the objects' bounding box in search crop coordinates. Afterwards, the predictions are stored in memory and transformed to image coordinates. The LSTM receives the respective search crop coordinates; the GraphCNN the image coordinates in contrast. Multiple objects are given to the network as a batch of elements, and hence, only the batch size increases with the number of objects. Figure 5.1 shows the structure of the network and details each layers' output size. We use ReLU activations for all layers detailed in the figure.

5.1 Long Short-Term Memory Module

We need a model that can encode previous motion information and can predict person-specific trajectories based on a dynamic length of historical movements. Recent work on the task of path prediction focuses mainly on the use of LSTM- and RNN-like structures to successfully predict a path based on previous measurements [96, 97, 98]. Nevertheless, most methods use one of such networks per individual. Due to the vast amount of objects in our scenarios, this is not applicable. We need a model that treats each object as an element in a mini-batch. Therefore, the number of models does not increase with the number of objects.

To test an LSTM's capability of predicting future movements of individual objects, we conduct a small experiment. We construct an LSTM consisting of two bidirectional LSTM layers and a dropout layer with $p = 0.5$ in between. We set the size of the hidden dimensions to 64. A final linear layer maps to an output of two numbers representing x and y values of

a movement vector. The input of the LSTM are two-dimensional movement vectors with a dynamic length history of up to five time steps. The results of this experiment show that this LSTM is capable of predicting the next movement vector to as accurate as 3.6 pixels, representing 0.43 m. Consequently, we do not consider it necessary to train one LSTM per object, but instead consider to train a single LSTM on multiple objects, giving the LSTM the ability to predict a movement vector based only on dynamic length historical movements of individuals.

We propose to use such an LSTM as a path prediction module in our network. Our LSTM module consists of two bidirectional LSTM layers. In a first step, the network generates a sequence of object motion vectors from its predictions. In our experiments, each track has a dynamic history of up to five last predictions. Tracks do not necessarily start at the same time, and consequently, the length of each track history can differ. As a consequence, we use zero-padding to achieve similar track history lengths, making it possible to process all tracks as a batch. These sequences are fed to the first LSTM layer with a hidden size of 64. We apply dropout with $p = 0.5$ to the hidden state of the first LSTM layer h_t^{l-1} and pass the result as input to the second LSTM layer. Afterwards, we feed the output features h_t^l of the second LSTM layer to a linear layer of size 128. The last step includes concatenating the output of the LSTM module Out_{LSTM} with Out_{SNN} and Out_{Graph}, the output of the GCNN module. The LSTM module allows the network to predict object locations based on a fusion of appearance and movement features.

5.2 GraphCNN Module

The GraphCNN module consists of three 1D convolution layers with 1×1 kernels, and the output channel numbers of 32, 64, and 128. We generate each object's adjacency graph based on the location prediction of all objects. The eight closest neighbors in a distance of 7.5 m of each object are considered and represented as directed graphs by a set of vectors with the object's position x, y as center node: $[x, y, vx_1, vy_1, ..., vx_8, vy_8]$. This graph contains the target object location in the image crop coordinates and the information of the vectors to the selected neighbors vx_i,vy_i. In case less than eight neighbors are existing, we zero-pad the respective vectors. The amount of previous graph configurations is limited to five, and similar to the LSTM module, we use zero-padding if less than 5 time steps are available. We feed the sequences of graphs as an input matrix of 18×5 to the first convolution layer. Multiple objects are given to the network as a batch of matrixes. We give the output features of the first layer to the next layers, upscaling channel size to 128 at the end of the convolutional block. We use global average pooling to generate the final output Out_{Graph} of this module consisting of 128 features. Similar to the LSTM module, we concatenate these features with Out_{SNN} and Out_{LSTM}. The GraphCNN module gives the network the ability to understand group movements better.

5.3 Experimental Setup

For all of our experiments, we used *PyTorch* and *Nvidia Titan XP* GPUs. We trained all networks with an SGD optimizer and an initial learning rate of 10^{-6}. For all training setups, unless indicated otherwise, we used the L1 loss, $L(x, y) = |x - y|$, where x and y are the output of the network and ground truth, respectively. The mini-batch size of all our experiments is 150; however, during offline feedback training, the batch size can differ due to unsuccessful tracking cases and the subsequent removal of the respective object from the batch.

During the training of SMSOT-CNN, we assigned different fractions of the initial learning rate to each layer, similar to its original implementation in *Caffe* inspired by the GOTURN's implementation. In detail, we assigned the initial learning rate to each convolutional layer while we assigned the same value multiplied by 10 to fully connected layers. Weight and bias initialization were also identical to the *Caffe* version, where weights were assigned as Gaussians with different standard deviations and biases as constants being zero or one. The training process of SMSOT-CNN is based on a so-called *Example Generator*. Provided with one target image with known object coordinates, it creates multiple examples by creating and shifting the search crop to create different kinds of movement. However, it is also possible to give the true target and search images. A hyperparameter controls the number of examples generated for each image, which was set to 10. We trained SMSOT-CNN for each dataset, respectively. However, for the pedestrian datasets, we used DLR-ACD to increase the amount of available training data. SMSOT-CNN is trained completely offline. Here, the network learns to regress the object location based on one time step only.

For AerialMPTNet, the SNN module and the fully connected layer were firstly trained similar to SMSOT-CNN. Afterwards, these layers were initialized with the obtained weights; the remaining layers were initialized with the standard *PyTorch* assignment. We decay the learning rate by a factor of 0.1 in every 20K iterations and train AerialMPTNet in an end-to-end fashion by using feedback loops to integrate previous movement and relationship information between adjacent objects. In contrast to the training process of SMSOT-CNN, which is based on artificial movement created by the example generator, we train our networks based on real tracks.

In the following, we describe the training process of our approach. In a first step, a batch of 150 random tracks (i.e., objects from random sequences of the training set) is selected starting at a random time step between 0 and the track end $t_{end} - 1$. We give the network a target and search crop for these objects, and the networks' goal is to regress the position in the search crops consecutively. However, in contrast to SMSOT-CNN, the training process works across multiple time steps. The object is left in the batch as long as the network tracks it successfully; consequently, target and search crop are based on the network predictions. However, if the true object position moves out of the predicted search area and the network fails, or the true track reaches an end, we remove the object from the batch and replace it with a new randomly selected object. For each track and each time step, the networks' prediction is stored and used from the LSTM and GCNN module. For each object in the batch, the LSTM module is given the objects' movement vectors from the last time steps up to 5. This process provides the

Name	SNN	LSTM	GraphCNN	SE Layers	OHEM
SMSOT-CNN	✓	×	×	×	×
AerialMPTNet$_{LSTM}$	✓	✓	×	×	×
AerialMPTNet$_{GCNN}$	✓	×	✓	×	×
AerialMPTNet	✓	✓	✓	×	×
AerialMPTNet$_{SE}$	✓	✓	✓	✓	×
AerialMPTNet$_{OHEM}$	✓	✓	✓	×	✓

Table 5.1: Visualization of different network settings.

network with an understanding of each object's motion characteristics and gives a prediction of the next movement. As a result, our network uses its predictions as feedback to improve its performance. Furthermore, we perform gradient clipping for the LSTM during training to prevent exploding gradients. The neighbor calculation of the GCNN module is also based on the networks' prediction of each object's position. Based on the network's output, we search for true nearest neighbors based on the entries of the training set in this specific sequence and frame. The neighbors are given to the module as a graph structure, also influencing the network's future prediction similar to the LSTM module. However, during the testing phase, we search nearest neighbors based on the network's prediction of the respective sequence and frame, not based on the ground truth.

For the pedestrian dataset, we chose a context factor of 4; hence, an object with a bounding box sized 4 x 4 pixel results in a 16 x 16 pixel sized crop. For vehicle tracking, however, we reduced the context factor to 3. The main reason is the large bounding box size, also resulting in a larger search window.

In Table 5.1, we visualized different tracking configuration. The table shows how the approaches differ from each other and introduces name conventions.

5.3.1 Squeeze-And-Excitation Layers

During our preliminary experiments, we experienced a high deviation in the quality of activation maps produced by the convolution layers in DCFNet and SMSOT-CNN. This deviation shows the direct impact of single channels and their importance for the final result of the network. Nevertheless, our network structure does not take this issue into account.

We decided to model the different importance of the single channels by using SE layers. We add one SE layer behind the SNN module, as well as one SE layer behind the weight-sharing multi-layer perceptrons. We visualize the resulting network architecture in Figure 5.2. The training process of the network does not change.

5.3.2 Online Hard Example Mining for Tracking

In the field of object detection, datasets contain an overwhelming number of easy samples and a small number of hard examples. Researchers developed several strategies to deal with

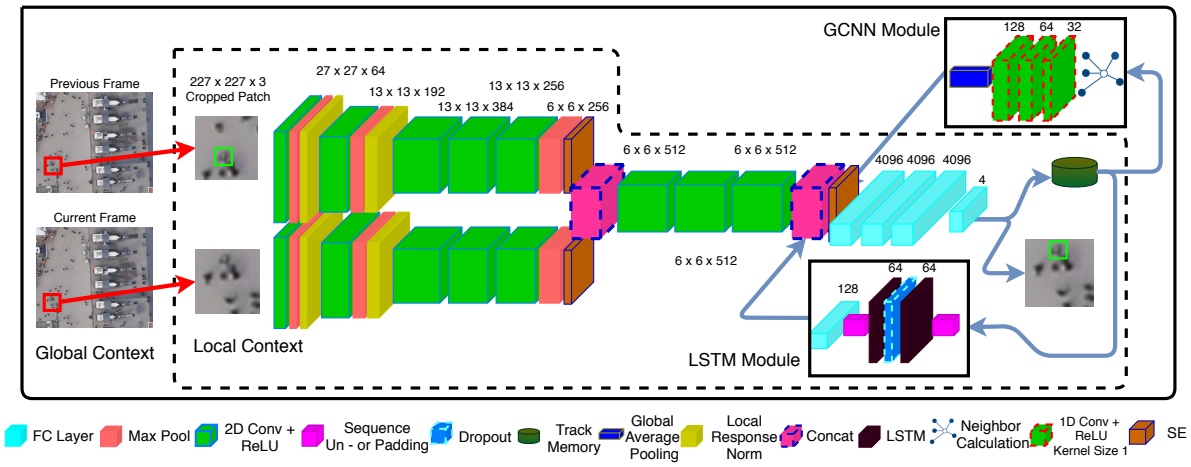


Figure 5.2: Overview of the network's architecture including SE layers.

this problem, such as special loss functions (e.g., Focal Loss [99]) penalizing the importance of easy examples, or hard example mining, which gives hard examples to the object detector it previously failed. The selection of such hard examples can make the training more effective.

However, in the field of multi-object tracking, such strategies are not commonly used. Nevertheless, datasets also contain a large number of easy samples and a smaller value of hard examples. To the best of our knowledge, none of the previous works in regression-based tracking used an online hard exampling mining (OHEM) strategy during training.

Hence, we propose to integrate OHEM in our training process in a simple way. If the tracker loses an object during training, we reset the object to its original start position and start frame, and give it to the network in the next iteration again. If the tracker fails again, we remove the sample from the mini-batch.

6 Evaluation & Discussion

In this chapter, we evaluate the proposed AerialMPTNet with multiple modifications on the KIT AIS datasets and the AerialMPT dataset. We compare our results to several methods such as KCF, Medianflow, MOSSE, CSRT, as well as DNN-based methods such as DeepSORT, Tracktor++, DCFNet, and SMSOT-CNN.

6.1 *PyTorch* SMSOT-CNN

SMSOT-CNN was originally written within the *Caffe* framework. However, we decided to adopt the code and move it to *PyTorch*. The usage of the saved weights and models was not possible. Hence, we trained a SMSOT-CNN model on each dataset. We visualized the respective results in Table 6.1. SMSOT-CNN achieves a MOTA and MOTP score of -35.0 and 70.0 for the KIT AIS pedestrian, and 37.1 and 75.8 for the KIT AIS vehicle dataset, respectively. Compared to the original implementation of SMSOT-CNN in Caffe, MOTA decreases by 5.2 and 4.0 points for the pedestrian and vehicle set, respectively. We report a MOTA and MOTP of -37.2 and 68.0 for the AerialMPT dataset.

6.2 AerialMPTNet (LSTM only)

We focused our first experiments on the LSTM-based AerialMPTNet. Table 6.2 demonstrates the tracking result of AerialMPTNet$_{LSTM}$ with frozen and trainable convolutional weights during training on the KIT AIS pedestrian dataset. We took the pretrained weights of SMSOT-CNN to initialize the convolutional weights and bias of both networks here.

The network with the unfrozen weights outperforms the one with the frozen weights. It reaches a MOTA score of -17.8 and a competitive MOTP of 68.8, while the frozen network's result is -26.0 and 69.3, respectively. Additionally, the unfrozen network achieves a significantly higher percentage of mostly tracked objects than the frozen network. It tracks 28.9 % of the pedestrians mostly, while the frozen network only tracks 22.0 % of them mostly. Furthermore, the unfrozen network achieves better results for IDF1, IDP, IDR, Rcll, Prcn, FAR, ML, FP, FN, and MOTAL. The improvement is distributed equally across all sequences. Nevertheless, it is striking that the amount of ID switches is 270 for the unfrozen network, compared to 231 for the frozen network. Based on a visual inspection, the lower amount of ID switches results from the loss of tracks in non-crossing situations, and thus, the tracker's estimate is placed on a space where no target appears. The unfrozen network holds trajectories better; however, the loosing of tracks happens mostly in more crowded situations leading to increased ID switches.

Table 6.1: Results of *PyTorch* SMSOT-CNN on KIT AIS and AerialMPT datasets.

KIT AIS Pedestrian Dataset

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	49.4	49.2	49.6	51.7	51.3	42.92	94	22.4	60.6	17.0	558	548	15	88	1.2	66.8	2.4
AA_Walking_02	17	29.6	29.0	30.2	31.9	30.6	113.76	188	9.1	45.7	45.2	1934	1820	25	139	-41.5	65.7	-40.6
Munich02	31	20.7	19.9	21.5	24.5	22.6	165.45	230	3.5	44.3	52.2	5129	4625	91	271	-60.7	67.1	-59.3
RaR_Snack_Zone_02	4	63.1	62.9	63.4	64.2	63.7	79.0	220	35.0	63.6	1.4	316	310	1	39	27.5	78.2	27.6
RaR_Snack_Zone_04	4	63.5	63.3	63.7	65.3	64.9	108.5	311	35.0	64.0	1.0	434	427	3	48	29.8	76.7	30.0
Total	69	32.5	31.7	33.4	35.7	33.9	121.32	1043	22.2	56.0	21.8	8371	7730	135	585	-35.0	70.0	-33.9

AerialMPT Dataset

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
Bauma3	16	29.3	28.6	30.0	34.6	33.0	385.69	609	9.9	47.1	43.0	6171	5748	200	458	-37.9	69.1	-35.7
Bauma6	26	30.8	28.6	33.3	37.7	32.3	161.23	270	12.2	57.4	30.4	4192	3311	115	302	-43.4	67.7	-41.2
Karlsplatz	27	30.7	29.4	32.2	33.8	30.8	94.93	146	6.9	58.2	34.9	2563	2233	26	95	-42.9	67.9	-42.2
Pasing7	24	57.7	54.5	61.3	61.9	55.1	43.42	103	35.9	54.4	9.7	1042	786	7	136	11.1	67.6	11.4
Pasing8	27	33.5	32.6	34.4	35.1	33.3	50.30	83	8.4	54.2	37.4	1358	1253	10	82	-35.7	67.0	-35.2
Witt	8	15.8	15.7	15.9	16.4	16.2	150.38	185	1.1	20.5	78.4	1203	1184	1	9	-68.6	61.5	-68.6
Total		32.0	30.7	33.4	36.6	33.6	129.13	1396	10.7	47.7	41.6	16529	14515	359	1082	-37.2	68.0	-35.6

KIT AIS Vehicle Dataset

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
MunichStreet02	20	87.4	85.0	90.1	90.5	85.3	5.80	47	87.2	8.5	4.3	116	71	1	7	74.8	80.6	74.9
StuttgartCrossroad01	14	67.3	63.6	71.5	74.9	66.6	14.86	49	57.1	30.6	12.3	208	139	3	17	36.8	75.3	37.3
MunichCrossroad02	45	50.6	49.5	51.7	53.5	51.3	24.38	66	45.5	27.3	27.2	1097	1001	17	41	1.9	69.4	2.6
MunichStreet04	29	83.5	82.4	84.7	85.8	83.6	8.83	68	76.5	14.7	8.8	256	215	6	15	68.6	79.7	68.9
Total	108	68.0	66.4	69.7	71.3	67.9	15.53	230	65.7	20.4	13.9	1677	1426	27	80	37.1	75.8	37.6

Table 6.2: Results of AerialMPTNet (LSTM only) on KIT AIS pedestrian dataset with pre-trained fixed and unfixed convolutional weights.

Fixed Weights

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	42.0	41.8	42.2	44.8	44.5	48.92	94	13.8	59.6	26.6	636	626	13	99	-12.3	68.4	-11.3
AA_Walking_02	17	34.7	34.0	35.4	37.2	35.8	104.94	188	8.0	55.3	36.7	1784	1678	22	227	-30.4	67.4	-29.7
Munich02	31	26.0	25.1	26.9	33.1	30.8	146.81	230	6.1	57.8	36.1	4551	4098	191	463	–44.3	67.8	-41.2
RaR_Snack_Zone_02	4	57.1	56.9	57.3	59.0	58.6	90.25	220	29.1	69.5	1.4	361	355	1	42	17.1	72.9	17.2
RaR_Snack_Zone_04	4	64.7	64.4	64.9	66.3	65.9	105.25	311	39.6	58.8	1.6	421	415	4	52	31.7	73.8	32.0
Total	69	35.5	34.6	36.3	40.4	38.5	112.36	1043	22.0	60.3	17.7	7753	7172	231	883	-26.0	69.3	-24.1

Unfixed Weights

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	47.1	49.9	47.3	49.6	49.2	44.77	94	23.4	48.9	27.7	582	572	11	91	-2.6	68.2	-1.8
AA_Walking_02	17	39.8	39.2	40.5	41.9	40.5	96.47	188	18.6	46.8	34.6	1640	1553	31	215	-20.7	67.2	-19.6
Munich02	31	29.6	28.6	30.8	37.1	34.5	139.10	230	8.3	59.6	32.1	4312	3852	221	506	-36.9	67.1	-33.3
RaR_Snack_Zone_02	4	63.0	62.8	63.2	64.9	64.4	77.50	220	37.3	60.0	2.7	310	304	4	31	28.6	72.2	28.9
RaR_Snack_Zone_04	4	67.6	67.5	67.8	69.1	68.8	96.50	311	46.0	50.8	3.2	386	380	3	43	37.5	73.3	37.7
Total	69	39.7	38.8	40.6	44.6	42.6	104.78	1043	28.9	53.8	17.3	7230	6661	270	886	-17.8	68.8	-15.5

Compared to the SMSOT-CNN baseline, the frozen and unfrozen AerialMPTNet$_{LSTM}$ achieves a total MOTA gain of 9 and 17.2 on the KIT AIS pedestrian dataset. MOTP decreases from 70.0 for SMSOT-CNN to 69.3 and 68.8 for the frozen and unfrozen AerialMPTNet$_{LSTM}$, respectively. We argue that the feature extracting SNN needs some finetuning to work well with the proposed LSTM module, which is the reason why the unfrozen networks show better results than the frozen one. Due to the advantages of trainable unfrozen weights, we used those for all of the remaining experiments. The MOTA enhancement of the unfrozen AerialMPTNet$_{LSTM}$ is especially high for the most complex sequences, *AA_Walking_02* and *Munich02*. It increases by 20.8 and 23.8 points, respectively. However, the MOTA decreases slightly to -2.6 from 1.2 for *AA_Crossing_02*. For the remaining sequences, it increases slightly.

In the next step, we also trained AerialMPTNet$_{LSTM}$ on the AerialMPT and the KIT AIS vehicle dataset. Table 6.3 shows the tracking results of AerialMPTNet$_{LSTM}$ on the AerialMPT

Table 6.3: Results of AerialMPTNet (LSTM only) on AerialMPT dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
Bauma3	16	28.3	27.7	29.0	34.6	33.0	386.00	609	8.4	51.2	40.4	6176	5745	246	608	-38.5	71.0	-35.7
Bauma6	26	33.2	31.2	35.5	39.3	34.5	152.35	270	13.0	58.5	28.5	3961	3225	135	387	-37.8	70.1	-35.3
Karlsplatz	27	48.4	47.0	50.0	51.4	48.2	68.89	146	24.7	55.5	19.8	1860	1641	16	140	-4.2	69.7	-3.8
Pasing7	24	61.0	58.5	63.6	64.3	59.2	38.08	103	35.9	56.3	7.8	914	737	5	127	19.8	70.5	20.0
Pasing8	27	41.3	40.6	42.1	42.7	41.4	43.78	83	18.1	50.6	31.3	1182	1108	4	90	-18.7	69.4	-18.6
Witt	8	15.6	15.5	15.7	17.3	17.1	148.75	185	2.7	23.8	73.5	1190	1171	3	24	-66.9	61.1	-66.8
Total	128	35.7	34.5	37.0	40.5	37.7	119.40	1396	12.8	49.8	37.4	15283	13627	409	1376	-28.1	70.1	-26.3

Table 6.4: Results of AerialMPTNet (LSTM only) on KIT AIS vehicle dataset.

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
MunichStreet02	20	81.9	79.9	84.0	84.9	80.6	7.60	47	74.5	10.6	14.9	152	113	4	3	63.9	79.6	64.4
StuttgartCrossroad01	14	65.9	62.4	69.9	72.7	65.0	15.50	49	59.2	26.5	14.3	217	151	2	11	33.2	76.2	33.5
MunichCrossroad02	45	57.7	56.0	59.5	60.6	56.9	21.93	66	48.5	33.3	18.2	987	850	22	43	13.7	69.4	14.7
MunichStreet04	29	88.7	88.3	89.1	89.9	89.0	5.79	68	86.8	7.4	5.8	168	153	2	3	78.7	79.8	78.8
Total	108	71.6	69.8	73.4	74.5	70.9	14.11	230	67.4	19.6	13.0	1524	1267	30	60	43.3	75.7	43.9

dataset. Similar to the KIT AIS pedestrian dataset, results improve significantly compared to the SMSOT-CNN baseline. The total MOTA score improves from -37.2 to -28.1, with MOTP also increasing from 68.0 to 70.1. The percentage of mostly tracked objects increases by 2.1 % to 12.8 %, the percentage of mostly lost tracks decreases by 4.2 % to 37.4 %. Additionally, the IDF1, IDP, IDR, Rcll, Prcn, FAR, PT, FPs, FNs, and MOTAL scores increase significantly compared to the baseline. The most complex sequences within AerialMPT are *Bauma3* and *Bauma6* captured at BAUMA trade fair, exhibiting overcrowded alleys and many pedestrian-intersection situations. MOTP increases slightly by 1.9 % and 2.4 %, MOTA decreases by 0.6 % and increases by 5.6 %. In such complex sequences, the LSTM module solely does not enforce the network with enough distinguishing power to discriminate against every single person. The *Karlsplatz* sequence achieves the largest MOTA improvement. It increases from -42.9 to -4.2 with MOTP, also increasing from 67.9 to 69.7. The results of the *Witt* sequence are mostly similar to the baseline, and we explain the reason for this behavior in section 6.4. The remaining sequences show a gradual improvement of most metrics.

Table 6.4 shows the quantitative results of AerialMPTNet$_{LSTM}$ on the KIT AIS vehicle dataset. The total MOTA and MOTP scores improved by 6.2 and 7.7 points to 43.3 and 75.7 compared to SMSOT-CNN, respectively. However, the performance gains are low compared to the results on the pedestrian datasets. IDF1, IDP, IDR, Rcll, Prcn, FAR, MT, ML, FP, FN, ID, FM, and MOTAL metrics improve significantly, with the number of ID switches reducing from 27 to 22. MOTA score on all sequences except *MunichStreet02* improves. We explain this behavior with visual inspections in section 6.4.

We conclude that trainable weights are important when adding new modules and improve the overall results significantly. Furthermore, the motion model encoded by the LSTM brings significant performance gains compared to the SMSOT-CNN baseline. AerialMPTNet$_{LSTM}$ outperforms SMSOT-CNN on all datasets in terms of MOTA and mostly tracked objects. Nevertheless, the improvement on the KIT AIS vehicle dataset is comparably low. We argue that the appearance features of vehicles are more dominant and easier to distinguish.

Table 6.5: Results of AerialMPTNet (GCNN only) on KIT AIS and AerialMPT datasets.

KIT AIS pedestrian

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	43.5	43.3	43.7	45.5	45.1	48.38	94	18.1	51.1	30.8	629	619	11	90	-10.9	68.5	-10.1
AA_Walking_02	17	35.8	35.3	36.2	38.2	37.2	101.35	188	14.9	47.9	37.2	1723	1650	35	204	-27.6	68.1	-26.3
Munich02	31	29.1	28	30.2	35.5	32.9	142.94	230	8.3	53.9	37.8	4431	3951	204	434	-40.2	68.1	-36.9
RaR_Snack_Zone_02	4	55.2	55.0	55.4	56.9	56.5	94.75	220	28.2	69.5	2.3	379	373	3	41	12.7	73.3	13.0
RaR_Snack_Zone_04	4	67.2	67	67.3	68.5	68.2	98.25	311	44.4	52.1	3.5	393	387	6	45	36.1	73.9	36.5
Total	69	37.5	36.7	38.4	42.0	40.0	109.49	1043	25.3	55.3	19.4	7555	6980	259	814	-23.0	69.6	-20.9

AerialMPT

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
Bauma3	16	29.6	28.9	30.4	36.5	34.7	376.75	609	11.3	48.3	40.4	6028	5581	276	550	-35.2	70.0	-32.1
Bauma6	26	36.7	34.4	39.3	43.7	38.2	144.23	270	20.4	50.4	29.2	3750	2994	126	329	-29.3	70.6	-26.9
Karlsplatz	27	43.7	72.3	45.2	46.4	43.4	75.63	146	15.8	63.0	21.2	2042	1809	25	145	-14.9	68.5	-14.2
Pasing7	24	68.6	66.0	71.4	71.6	66.1	31.50	103	51.5	39.8	8.7	756	857	4	96	34.7	71.0	34.9
Pasing8	27	41.2	40.4	42.1	42.7	41.0	44.0	83	18.1	51.8	30.1	1188	1108	2	94	-18.9	68.2	-18.9
Witt	8	14.1	14.0	14.2	15.3	15.1	152.38	185	1.6	19.5	78.9	1219	1200	0	15	-70.8	60.8	-70.8
Total	128	37.0	35.7	38.3	42.0	39.1	117.05	1396	15.6	46.0	38.4	14983	13279	433	1229	-25.4	69.7	-23.5

KIT AIS Vehicle

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
MunichStreet02	20	82.6	80.5	84.7	85.4	81.1	7.40	47	76.6	6.4	17.0	148	109	4	3	65.0	79.5	65.5
StuttgartCrossroad01	14	70.0	66.5	73.8	76.7	69.1	13.57	49	65.3	22.4	12.3	190	129	2	11	42.1	75.7	42.3
MunichCrossroad02	45	56.3	54.7	58.0	59.4	56.0	22.33	66	44.0	34.8	21.2	1005	876	14	41	12.1	70.0	12.7
MunichStreet04	29	87.3	86.8	87.8	88.5	87.4	6.66	68	83.8	8.8	7.4	193	175	2	3	75.6	79.7	75.7
Total	108	71.1	69.4	72.9	74.1	70.6	14.22	230	67.0	18.7	14.3	1536	1289	22	58	42.8	75.9	43.2

6.3 AerialMPTNet (GCNN only)

We focus our second experiments on adjacent neighbor modeling with AerialMPTNet$_{GCNN}$. We trained one model for each dataset. Table 6.5 shows the individual results for the KIT AIS pedestrian and vehicle datasets, as well as for the AerialMPT dataset.

The results of the KIT AIS pedestrian dataset improve significantly compared to SMSOT-CNN. AerialMPTNet$_{GCNN}$ reaches a total MOTA and MOTP of -23.0 and 69.6, which is an improvement of 12.0 and a decrease of 0.4, respectively. The tracker tracks 25.3 % of all pedestrians mostly and only looses 19.4 % of them mostly. This result is an improvement of 3.1 % and -2.4 %, respectively. The IDF1, IDP, IDR, Rcll, Prcn, FAR, MT, ML, FP, FN, and MOTAL scores surpass the previous results and prove that the knowledge of relationships between adjacent objects gradually increases the tracker's performance. As we have seen previously, the most complex sequences show the greatest improvement. This is also valid for this experiment: MOTA on *AA_Walking_02* and *Munich02* increase by 13.9 and 20.5, while MOTP climbs by 2.4 and 1.0. However, MOTA decreases by 12.1 and 14.8 on *AA_Crossing_02* and *RaR_Snack_Zone_02*. We argue that these sequences are less crowded, resulting in graph padding, which harms the tracker's performance. Compared to AerialMPTNet$_{LSTM}$, the total MOTA decreases by 5.2, and the total MOTP increases by 0.8.

The results of AerialMPTNet$_{GCNN}$ on the AerialMPT dataset also indicate that neighbor modeling is an important step to increase the tracker's performance. The total MOTA and MOTP scores surpass the results of the SMSOT-CNN baseline by 11.8 and 1.7, resulting in total values of -25.4 and 69.7, respectively. The percentage of mostly tracked pedestrians climb from 10.7 % to 15.6 %, while the percentage of mostly lost tracks reduces from 41.6 % to 38.4 %. Similar to our previous experiments, the IDF1, IDP, IDR, Rcll, Prcn, FAR, MT, ML. FP, FN, and MOTAL metrics improve. However, in contrast to the KIT AIS pedestrian dataset,

AerialMPTNet$_{GCNN}$ performs better than AerialMPTNet$_{LSTM}$. Based on the characteristics of the datasets, we explain this behavior with the higher crowd density in AerialMPT, making relationships between neighbors more critical than their movements.

The total results of AerialMPTNet$_{GCNN}$ trained on the KIT AIS vehicle dataset also improve compared to SMSOT-CNN. The total MOTA and MOTP climb by 5.7 and 0.1, resulting in a score of 42.8 and 75.9, respectively. Similar to AerialMPTNet$_{LSTM}$, the MOTA score of all sequences except *MunichStreet02* improves. However, it is striking that the results of *MunichCrossroad02* are comparably low. Visual inspections in section 6.4 will explain this behavior. Compared to AerialMPTNet$_{LSTM}$, the total MOTA decreases by 0.8, and the total MOTP increases by 0.4. In general, both networks have comparable results on this dataset.

6.4 AerialMPTNet

Our third experiment aims to unite the advantages of the LSTM and the GraphCNN module. We train a model of AerialMPTNet on each dataset.

Table 6.6 summarizes the results of AerialMPTNet on each dataset. AerialMPTNet surpasses the performance of AerialMPTnet$_{LSTM}$ and AerialMPTNet$_{GCNN}$ in terms of the most metrics for both pedestrian datasets. However, for the vehicle dataset AerialMPTNet falls behind both previous versions.

On the KIT AIS pedestrian dataset, the total MOTA increases by 1.6 and 6.8, respectively, to -16.2. The total MOTP moves +0.8 and 0.0 points to 69.6. Figure 6.1 shows a precision and a MOTA plot with different association IoU thresholds, comparing AerialMPTNet with its SMSOT-CNN baseline. The plots prove that AerialMPTNet permanently outperforms SMSOT-CNN. The network tracks 28.1 % of the pedestrians mostly, while it only looses 16.6 % mostly. Compared to the LSTM- and GCNN based networks, ML score improves by -0.7 % and -2.8 %. We also visualized MT and ML plots in Figure 6.2. Similar to the MOTA and precision plots, AerialMPTNet surpasses the performance of SMSOT-CNN at almost any threshold. Nevertheless, SMSOT-CNN loses fewer tracks mostly, when the threshold is set to 0.8 and 0.9. The reason for this behavior lies in the MOTP score, which is slightly worse for AerialMPTNet (69.6 to 70.0).

AerialMPTNet reaches the best scores among all previously tested methods on *AA_Walking_02*, *Munich02* and *RaR_Snack_Zone_02*, with MOTA scores of -16.8, -34.5 and 38.9. These sequences are the most complex sequences considering their lengths as well as their number of pedestrians and movements. The sequence length impacts the MOTA scores gradually, with longer sequences achieving lower scores than shorter ones. We show the illustrative results of SMSOT-CNN and AerialMPTNet on the *AA_Walking_02* sequence in Figure 6.3. White dots visualize the ground truth object positions, while blue dots are the tracker's estimates. The amount of visible white dots indicates that SMSOT-CNN loses many pedestrians within this sequence. Blue dots are stuck at the lines on the ground and do not move anymore within the visualized frames. The lines share similar appearance features with some pedestrians, and SMSOT-CNN cannot handle this problem. For AerialMPTNet, the amount of visible white dots decreased significantly, tracking the pedestrians crossing the lines successfully. On the

Table 6.6: Results of AerialMPTNet on the KIT AIS and AerialMPT datasets.

KIT AIS pedestrian

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
AA_Crossing_02	13	46.7	45.6	46.9	49.3	48.8	45.08	94	23.4	51.1	25.5	586	576	12	92	-3.4	69.7	-2.5
AA_Walking_02	17	41.4	40.8	42.1	43.7	42.3	93.59	188	17.0	51.6	31.4	1591	1504	25	231	-16.8	68.5	-15.9
Munich02	31	31.2	30.2	32.3	37.8	35.3	136.77	230	10.4	55.7	33.9	4240	3808	192	498	-34.5	67.6	-31.4
RaR_Snack_Zone_02	4	59.0	58.8	59.2	60.9	60.5	86.00	220	33.2	65.0	1.8	344	3338	4	34	20.7	73.4	21.1
RaR_Snack_Zone_04	4	68.5	68.3	68.6	69.8	69.5	94.25	311	45.7	51.8	2.5	377	371	3	42	38.9	74.2	39.1
Total	69	40.6	39.7	41.5	45.1	43.2	103.45	1043	28.1	55.3	16.6	7138	6597	236	897	-16.2	69.6	-14.2

AerialMPT

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
Bauma3	16	31.2	30.4	32.0	38.2	36.3	368.12	606	11.6	51.7	36.7	5890	5435	277	582	-32.0	70.8	-28.9
Bauma6	26	37.2	34.8	39.9	44.2	38.6	143.69	270	17.0	58.1	24.9	3736	2964	123	333	-28.4	70.2	-26.1
Karlsplatz	27	45.6	44.2	47.1	48.6	45.6	72.37	146	19.9	61.6	18.5	1954	1733	25	153	-10.0	67.4	-9.3
Pasing7	24	67.6	64.8	70.7	71.3	65.3	32.58	103	49.5	43.7	6.8	782	593	5	93	33.1	70.7	33.3
Pasing8	27	39.7	38.7	40.8	41.3	39.2	45.85	83	15.7	55.4	28.9	1238	1134	2	83	-22.9	68.9	-22.8
Witt	8	16.0	15.9	16.1	17.9	17.6	147.75	185	2.7	24.3	73.0	1182	1163	4	25	-65.9	60.1	-65.7
Total	128	37.8	36.5	39.3	43.1	40.0	115.48	1396	15.3	49.9	34.8	14782	13022	436	1269	-23.4	69.7	-21.5

KIT AIS vehicle

Evaluation	# Images	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
MunichStreet02	20	83.2	81.1	85.4	86.3	82.0	07.05	47	76.6	10.6	12.7	141	102	4	3	66.9	80.1	67.3
StuttgartCrossroad01	14	68.4	65.0	72.2	75.3	67.8	14.14	49	61.2	26.5	12.3	198	137	1	16	39.4	76.3	39.5
MunichCrossroad02	45	54.5	52.9	56.3	58.5	54.9	22.96	66	43.9	37.9	18.2	1033	895	20	45	9.6	70.1	10.5
MunichStreet04	29	86.5	86.0	87.0	89.1	88.0	6.34	68	85.3	7.4	7.3	184	165	4	3	76.8	80.2	77.0
Total	108	70.0	68.3	71.8	73.9	70.3	14.41	230	66.5	20.9	12.6	1556	1299	29	67	42.0	76.3	42.6

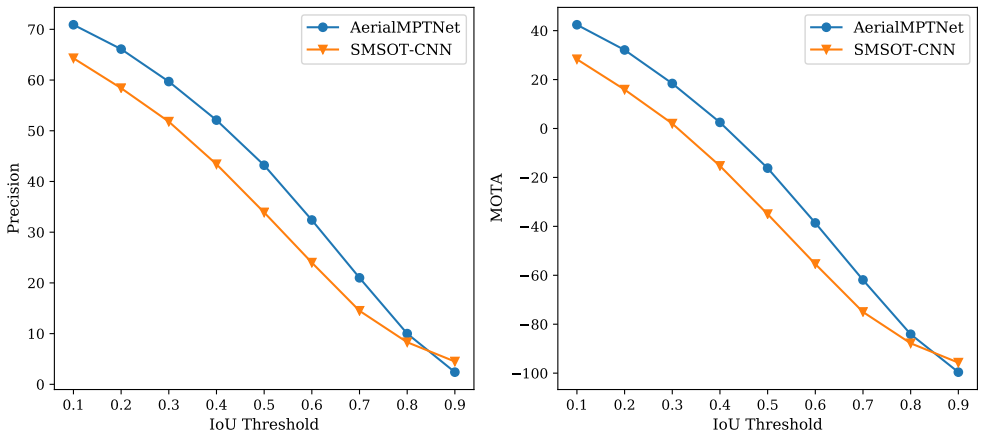


Figure 6.1: MOTA and Precision plots, showing the results of AerialMPTNet and its baseline SMSOT-CNN on the KIT AIS pedestrian dataset. AerialMPTNet outperforms SMSOT-CNN at almost any IoU threshold.

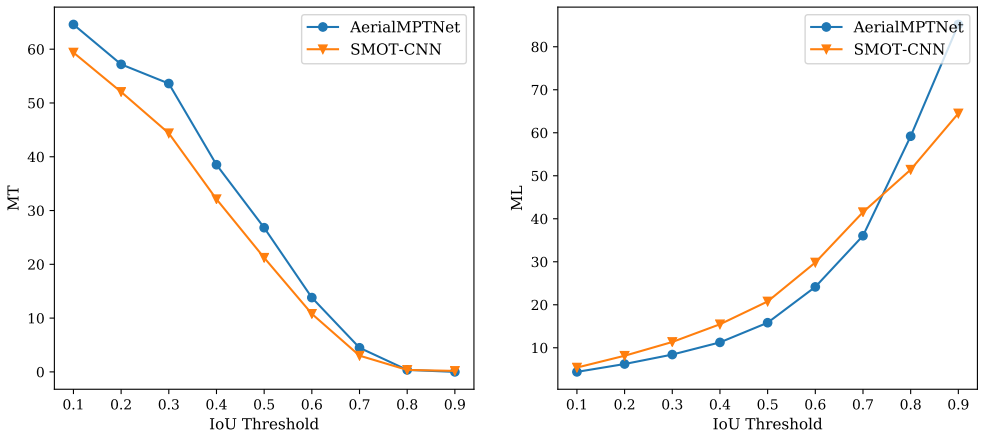


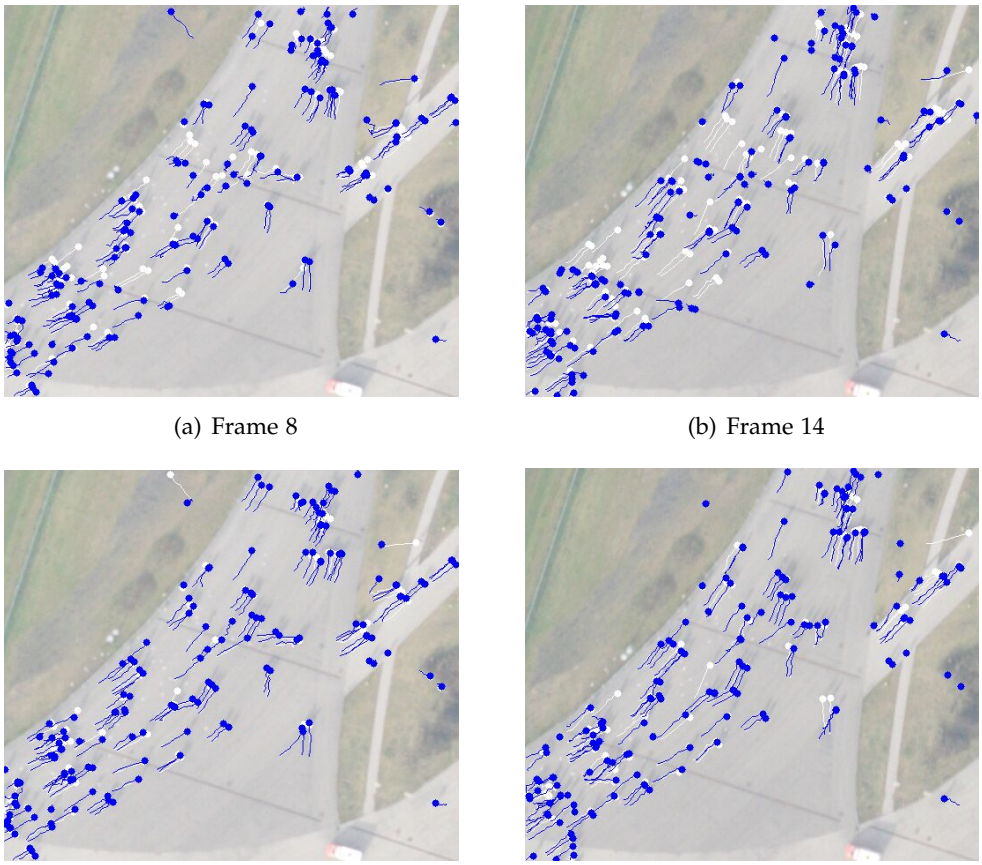
Figure 6.2: MT and ML plots, showing the results of AerialMPTNet and its baseline SMSOT-CNN on the KIT AIS pedestrian dataset. AerialMPTNet outperforms SMSOT-CNN at almost any IoU value. However, it strikes that the ML score of AerialMPTNet increases significantly for IoU threshold greater than 0.7.

remaining sequences, AerialMPTNet achieves competitive performance. We also visualized a cropped part of the *AA_Crossing_02* sequence in Figure 6.4. Similar to *AA_Walking_02*, AerialMPTNet tracks the pedestrians crossing the line successfully; however, it loses the object located on the right-side grass area that SMSOT-CNN tracks successfully.

AerialMPTNet follows a similar trend on the AerialMPT dataset. It outperforms the previous LSTM- and GCNN-based methods by 4.7 and 2.0 in terms of MOTA, respectively, while holding a competitive MOTP of 69.7. We visualized the MOTA and precision plots in Figure 6.5. Similar to KIT AIS pedestrian, AerialMPTNet outperforms SMSOT-CNN for any threshold. MOTA and precision are continuously higher for AerialMPTNet, proving the effectiveness of the LSTM and GraphCNN module. AerialMPTNet tracks 15.3 % of the pedestrians mostly and looses 34.8 % of them mostly. Compared to AerialMPTNet$_{LSTM}$ and AerialMPTNet$_{GCNN}$, it looses 2.6 % and 3.6 % pedestrians less, respectively. Figure 6.6 shows the MT and ML plots of AerialMPTNet comparing with the SMSOT-CNN baseline. AerialMPTNet outperforms SMSOT-CNN at any threshold. The MOTA score increases for every single sequence compared to SMSOT-CNN. Additionally, AerialMPTNet achieves the best MOTA score among all previously tested methods on the *Bauma3*, *Bauma6* and *Witt* sequences (-32.0, -28.4, -65.9), which contain the most complex scenarios within the AerialMPT dataset, regarding crowd density and movement, variety of GSDs, as well as the complexity of the terrain.

However, in contrast to the KIT AIS pedestrian dataset, the MOTA scores are not correlated with the sequence lengths. This observation proves that the scene complexities are well-distributed over the different sequences of AerialMPT.

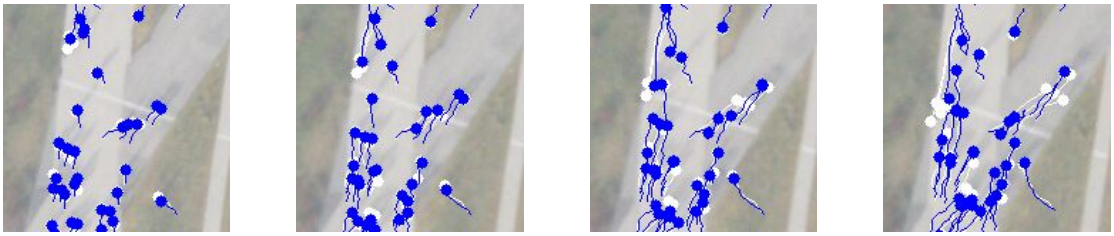
In Figure 6.7, we show the tracker's ability to encode motion with the LSTM module,



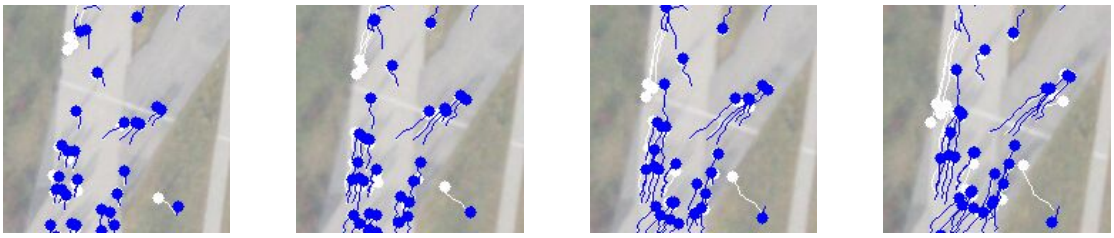
(a) Frame 8 (b) Frame 14

(c) Frame 8 (d) Frame 14

Figure 6.3: Illustrative results of SMSOT-CNN (top) and AerialMPTNet (bottom) on sequence *AA_Walking_02*. White dots symbol ground truth positions, blue dots the tracker's estimates. SMSOT-CNN looses tracks due to the ground terrain sharing similar appearance features with the pedestrians. AerialMPTNet outperforms SMSOT-CNN significantly.



(a) Frame 4 (b) Frame 6 (c) Frame 8 (d) Frame 10



(e) Frame 4 (f) Frame 6 (g) Frame 8 (h) Frame 10

Figure 6.4: Comparing the performance of AerialMPTNet (bottom) and SMSOT-CNN (top) on sequence AA_Crossing_02. White dots symbol ground truth positions, blue dots the tracker's estimates. In contrast to SMSOT-CNN, AerialMPNet holds the trajectory of the pedestrians crossing the line on the ground well; however, it looses track of the pedestrian walking on grass that SMSOT-CNN tracks successfully.

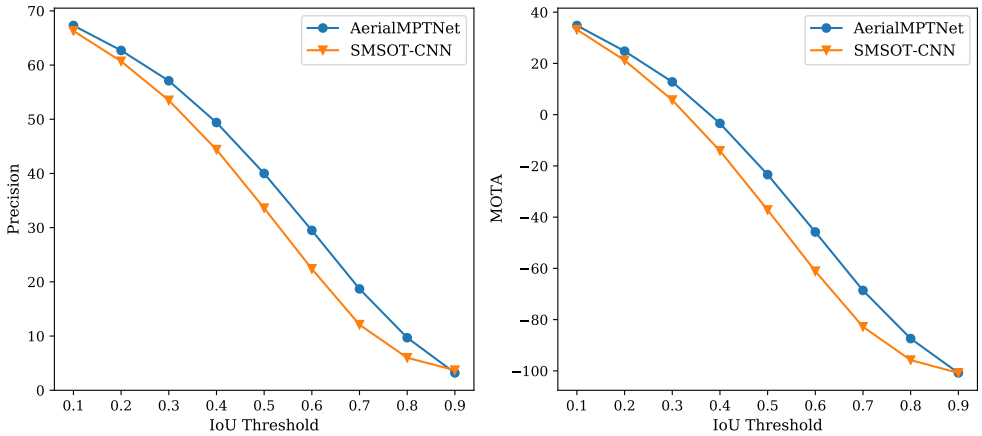


Figure 6.5: MOTA and Precision plots, showing the results of AerialMPTNet and its baseline SMSOT-CNN on the AerialMPT dataset. AerialMPTNet outperforms SMSOT-CNN at almost any IoU value.

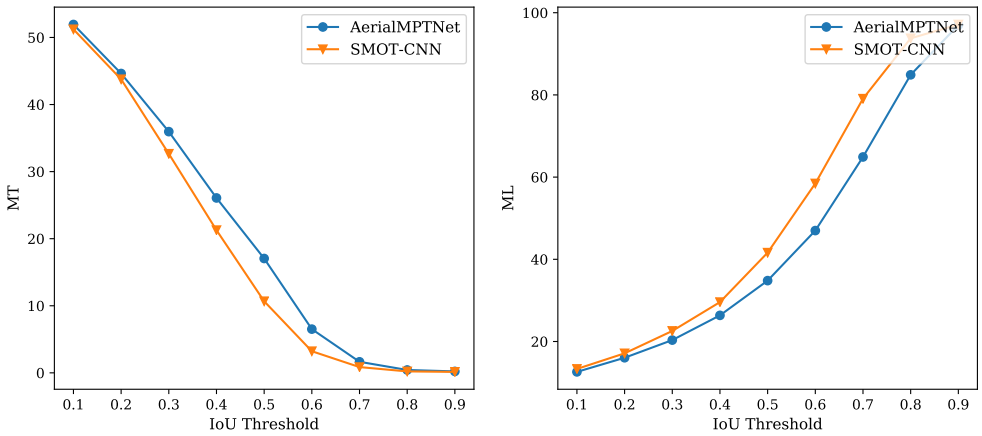


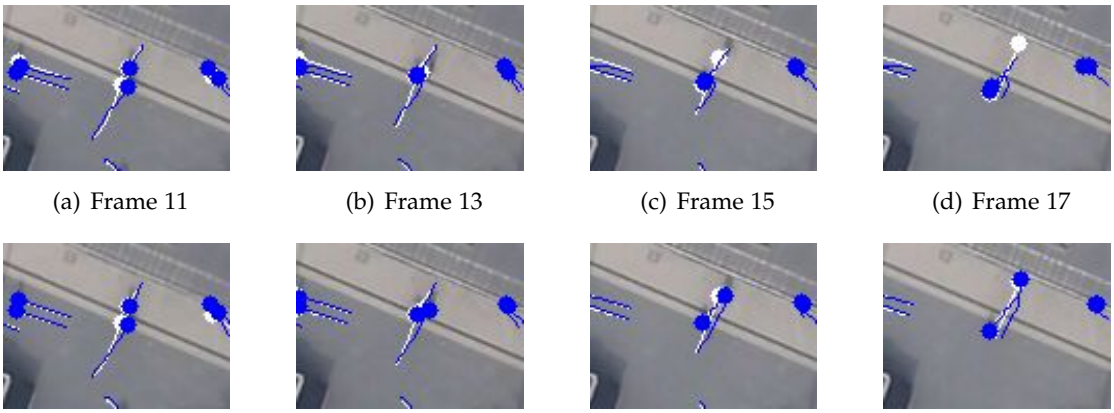
Figure 6.6: MT and ML plots, showing the results of AerialMPTNet and its baseline SMSOT-CNN on the AerialMPT dataset. AerialMPTNet outperforms SMSOT-CNN at almost any IoU value

with crops taken from the *Pasing-8* sequence. The top row shows the illustrative results of SMOT-CNN. It looses one pedestrian within the intersection scenario, leading to an ID switch. AerialMPTNet, in the bottom row, tackles this challenge successfully and can track both pedestrians correctly. The fusion of the appearance with temporal and graphical features allows AerialMPTNet to outperform the SMSOT-CNN baseline by better handling the pedestrian crossing situations and keeping the pedestrian trajectories over a longer term even in the presence of interrupting features.

Furthermore, visual inspections also prove the effectivity of the adjacent neighbor modeling with the GraphCNN module. We visualized some illustrative results of SMSOT-CNN (top) and AerialMPTNet (bottom) in Figure 6.8. The images are taken from the right bottom side of the *Karlsplatz* sequence and show a pedestrian crowd moving. While SMSOT-CNN has difficulties in tracking the pedestrians in the middle of the crop correctly, AerialMPTNet performs significantly better. We argue that that the observed improvement is mainly caused by our GraphCNN module.

Nevertheless, we find that there are also sequences where both networks reach their limits and perform poorly. Figure 6.9 shows the illustrative results of SMSOT-CNN (top) and AerialMPTNet (bottom) on the *Witt* sequence. The amount of visible white dots signals that both networks lose most pedestrians mostly. According to the results, despite its small number of frames, the MOTA score is also low for both trackers (-68.6 vs. -65.9). Further investigation shows that a non-adaptive search window size causes this poor performance. In the Witt sequence, objects move out of the search window and are lost by the tracker. Varying GSDs and pedestrian speeds need to be taken into account more intensively to solve these problems.

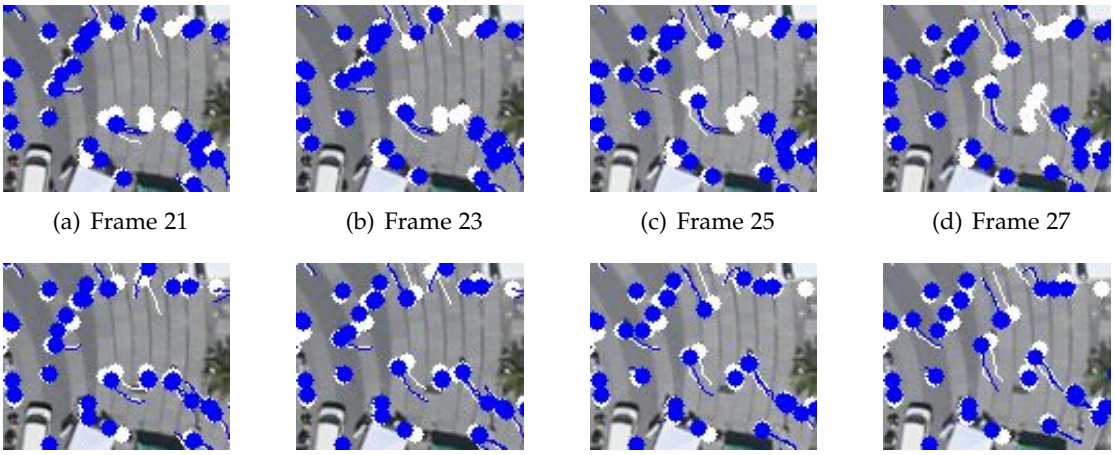
To conclude the evaluation of the pedestrian datasets, we show AerialMPTNet estimates



(a) Frame 11 (b) Frame 13 (c) Frame 15 (d) Frame 17

(e) Frame 11 (f) Frame 13 (g) Frame 15 (h) Frame 17

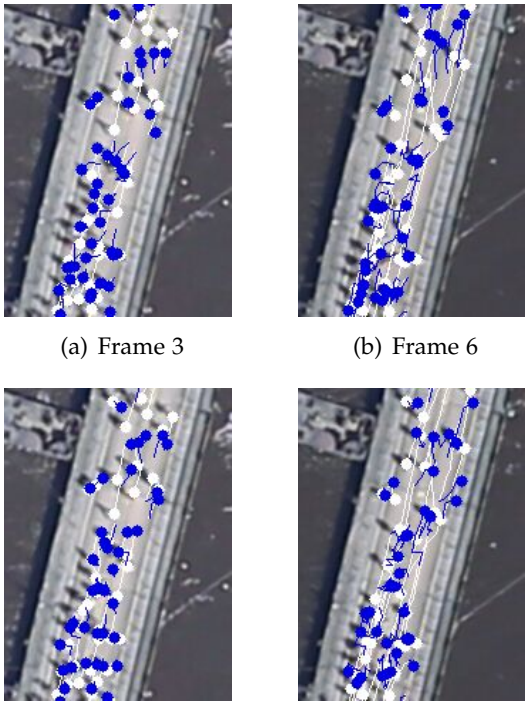
Figure 6.7: Comparing the performance of AerialMPTNet (bottom) and SMSOT-CNN (top) on sequence Pasing-8. White dots symbol ground truth positions, blue dots the tracker's estimates. AerialMPTNet's LSTM encodes a learned motion model and keeps track of trajectories even in pedestrian intersections. SMSOT-CNN cannot handle such intersections well.



(a) Frame 21 (b) Frame 23 (c) Frame 25 (d) Frame 27

(e) Frame 21 (f) Frame 23 (g) Frame 25 (h) Frame 27

Figure 6.8: Comparing the performance of AerialMPTNet (bottom) and SMSOT-CNN (top) on sequence Karlsplatz. White dots symbol ground truth positions, blue dots the tracker's estimates. AerialMPTNet outperforms SMSOT-CNN clearly, and holds trajectories significantly better in crowded situations.



(a) Frame 3 (b) Frame 6

(c) Frame 3 (d) Frame 6

Figure 6.9: Comparing the performance of AerialMPTNet (bottom) and SMSOT-CNN (top) on sequence *Witt*. White dots symbol ground truth positions, blue dots the tracker's estimates. Both trackers show a poor performance since pedestrians move out of their search window too quickly.

for frame 18 and 10 of the *Munich02* and *Bauma3* sequences in Figure 6.10, respectively. The images give an overview of AerialMPTNet's performance and show the difficulties it can handle.

The results on the KIT AIS vehicle dataset are also promising and outperform those of SMSOT-CNN; however, the performance gains are low compared to the pedestrian results. AerialMPTNet reaches a total MOTA and MOTP of 42.0 and 76.3. Compared to the SMSOT-CNN baseline, this is an increase of +4.9 and +9.5. We visualized the MOTA and precision plots in Figure 6.11. AerialMPTNet outperforms SMSOT-CNN in terms of MOTA and precision at any threshold, however, by a lower margin compared to pedestrian tracking. The LSTM-based network reaches a MOTA of 43.3, and the GCNN-based network a MOTA of 42.8, and hence they perform better solely than the combined AerialMPTNet. This effect can be seen among most of the metrics. It results from the construction of the network itself, which we fitted gradually for pedestrian tracking. Consequently, the thresholds we set may not be suitable for vehicle tracking. A good example is the distance threshold for neighbor modeling, where only objects within a distance of 50 pixels from the target object were considered. For vehicle tracking, such settings differ, but finding a more suitable threshold was out of the scope of this thesis. AerialMPTNet tracks 67.0 % of the vehicles mostly and loses only 14.3 % of the vehicles. The plots comparing SMSOT-CNN and AerialMPTNet regarding ML and MT are visible in Figure 6.12. In contrast to the previous plots, AerialMPTNet does not perform significantly better than SMSOT-CNN. For the ML metrics, SMSOT-CNN even performs slightly better continuously.

Looking at the respective vehicle sequences, it is striking that results of *MunichCrossroad02* are meager, with the MOTA 29.8 points worse than on the second-worst *StuttgartCrossroad01*sequence. Figure 6.13 visualizes the problems AerialMPTNet has to deal with in this sequence. *MunichCrossroad02* includes strong camera movement, which affects scene arrangement and appearance. Additionally, harsh shadows and trees cover vehicles partly. These effects, combined with intricate movement patterns, lead to the poor performance of AerialMPTNet on this sequence.

Figure 6.14 and Figure 6.15 show the outperformance of AerialMPTNet compared to SMSOT-CNN. The difference is especially visible in Figure 6.15, where AerialMPTNet tracks the truck in the middle of the pictures significantly better.

However, according to Table 6.6, SMSOT-CNN produces better results on the *MunichStreet02* sequence. Based on visual inspection in Figure 6.16, we see that AerialMPTNet loses one vehicle in the middle of the picture, which SMSOT-CNN can track successfully. However, we argue that these problems can be solved by adjusting the parameter settings accordingly.

6.4.1 Additional Experiments

In this subsection, we evaluate additional experiments, including SE layers, different loss functions, and OHEM.

We changed our network architecture as described in subsection 5.3.1, and added SE layers. We trained one model for each dataset, and compare the quantitative results with AerialMPTNet in Table 6.7. For both of the KIT AIS dataset, overall results worsen, with the

(a) Frame 18

(b) Frame 10

Figure 6.10: Illustrative predictions of AerialMPTNet on *Munich02* and *Bauma3* sequence.

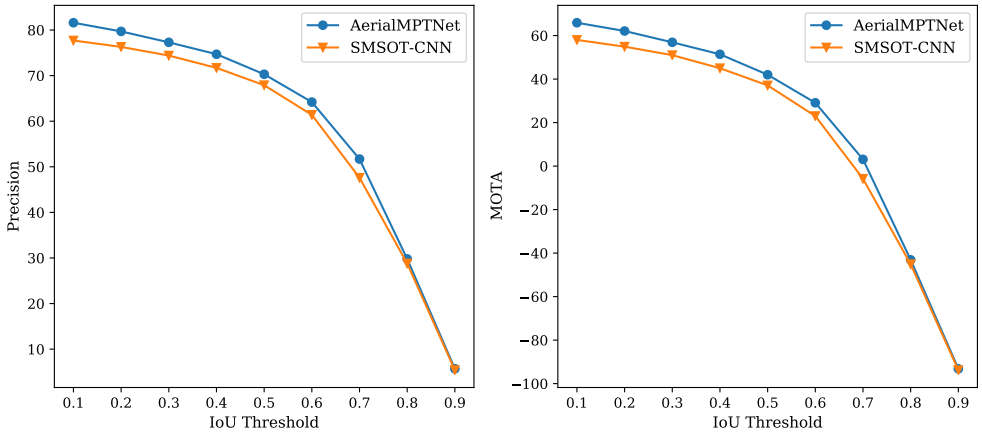


Figure 6.11: MOTA and Precision plots, showing the results of AerialMPTNet and its baseline SMSOT-CNN on the KIT AIS vehicle dataset. AerialMPTNet outperforms SMSOT-CNN at any IoU threshold.

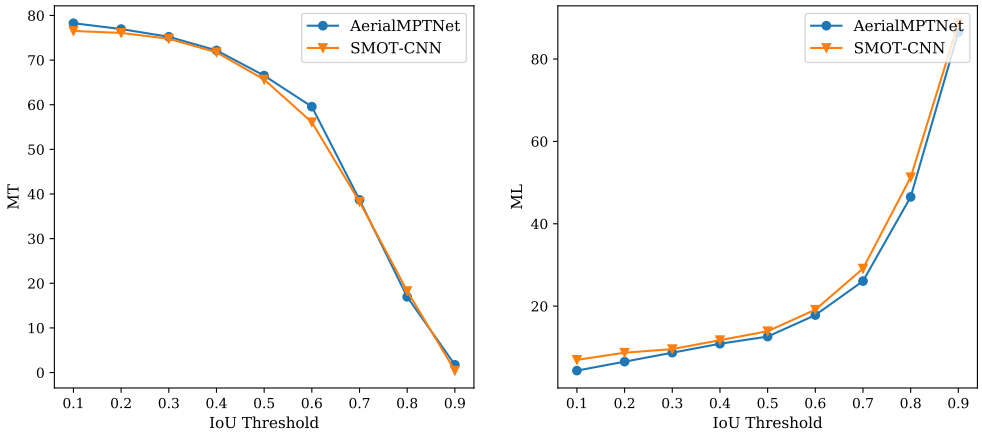
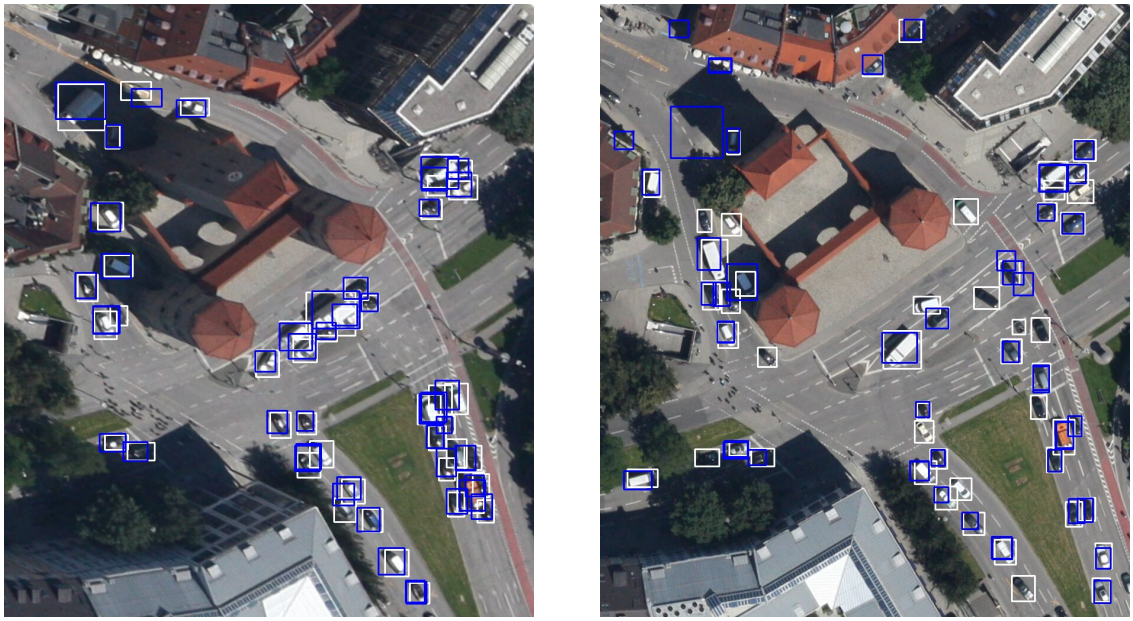
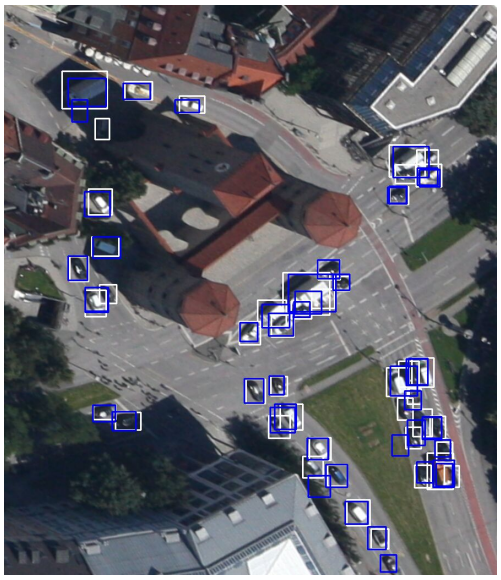


Figure 6.12: MT and ML plots, showing the results of AerialMPTNet and its baseline SMSOT-CNN on the KIT AIS vehicle dataset. AerialMPTNet outperforms SMSOT-CNN at any IoU threshold.

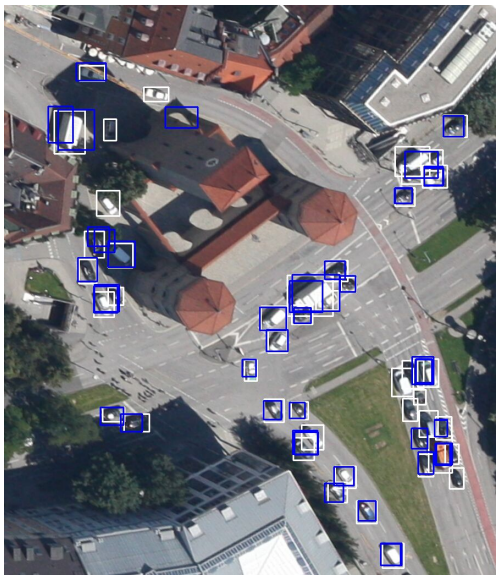


(a) Frame 4 (b) Frame 31

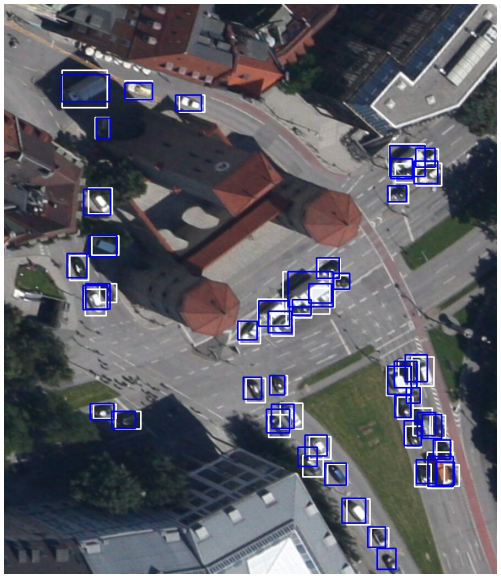
Figure 6.13: Visualization of the difficulties AerialMPTNet faces. The camera configuration and scene appearance change continously. Additinoally, shawdows and trees reduce visibility.



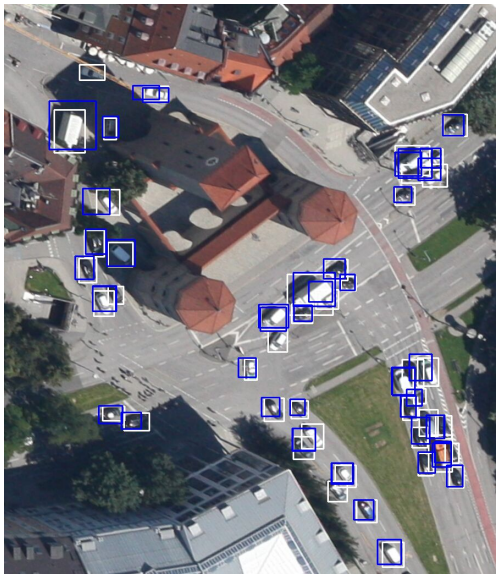
(a) Frame 2



(b) Frame 8

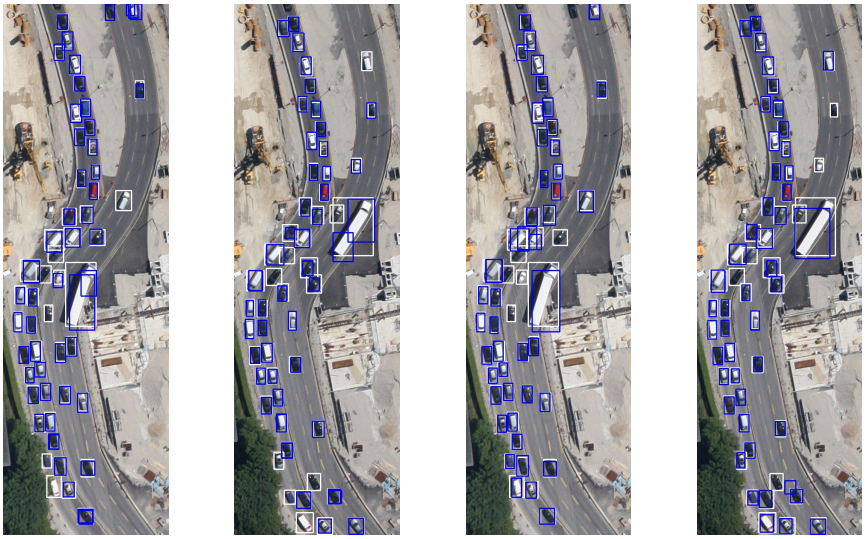


(c) Frame 2



(d) Frame 8

Figure 6.14: Comparing the performance of AerialMPTNet (bottom) and SMSOT-CNN (top) on sequence *MunichCrossroad02*. White bounding boxes symbol ground truth positions, blue bounding boxes the tracker's estimates. In contrast to SMSOT-CNN, AerialMPTNet holds the trajectories slightly better.



(a) Frame 20 (b) Frame 29 (c) Frame 20 (d) Frame 29

Figure 6.15: Comparing the performance of AerialMPTNet (right) and SMSOT-CNN (left) on sequence *MunichStreet04*. White bounding boxes symbol ground truth positions, blue bounding boxes the tracker's estimates. AerialMPTNet achieves better results for this sequence. For example, AerialMPTNet tracks the truck in the middle of the pictures without any additional bounding box. SMSOT-CNN switches ID there.

(a) Frame 1 (b) Frame 7 (c) Frame 1 (d) Frame 7

Figure 6.16: Comparing the performance of AerialMPTNet (right) and SMSOT-CNN (left) on sequence *MunichStreet02*. White bounding boxes symbol ground truth positions, blue bounding boxes the tracker's estimates. SMSOT-CNN generates better estimates within this sequence.

Table 6.7: Comparison of AerialMPTNet and AerialMPTNet$_{SE}$.

Dataset	SE	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS pedestrian	No	**40.6**	**39.7**	**41.5**	**45.1**	**43.2**	**103.45**	1043	**28.1**	**55.3**	**16.6**	**7138**	**6597**	**236**	897	**-16.2**	69.6	**-14.2**
KIT AIS pedestrian	Yes	38.3	37.5	39.1	42.8	41.1	107.17	1043	27.4	54.5	18.1	7395	6876	250	**818**	-20.7	**69.9**	-18.7
AerialMPT	No	37.8	36.5	39.3	43.1	40.0	115.48	1396	15.3	**49.9**	**34.8**	14782	13022	436	1269	-23.4	69.7	-21.5
AerialMPT	Yes	**38.9**	**37.5**	**40.4**	**44.1**	**40.9**	**113.81**	1396	**17.0**	48.1	34.9	**14568**	**12799**	**430**	**1212**	**-21.4**	**69.8**	**-19.6**
KIT AIS vehicle	No	**70.0**	68.3	**71.8**	**73.9**	**70.3**	**14.41**	230	**66.5**	20.9	12.6	**1556**	**1299**	29	**67**	**42.0**	**76.3**	**42.6**
KIT AIS vehicle	Yes	**70.0**	**68.4**	71.7	73.2	69.8	14.57	230	63.5	**24.8**	**11.7**	1574	1334	**23**	84	41.1	75.6	41.5

MOTA falling from -16.2 to -20.7 for the pedestrian set and from 42.0 to 41.1 for the vehicle part. The Rcll, Prcn, FAR, IDR, MT, FP, FN, and MOTAL score also worsen compared to AerialMPTNet for both datasets. However, for KIT AIS vehicle, we can see an improvement of the ML score from 12.6 % to 11.7 %.

We observe a completely different behavior for the AerialMPT dataset. All metrics, except PT and ML, show a gradual improvement with Squeeze-and-Excitation layers. MOTA climbs from -23.4 to -21.4, and even the MOTP climbs 0.1 points to 69.8. The percentage of mostly tracked objects increases by 1.7 % to 17.0 %.

It is difficult to judge these results since they differ so much. However, we argue that the differing quality of the images influences the results. The images in AerialMPT have significantly better contrast, and hence, the adaptive weighting of channels is more purposeful.

Furthermore, we also experimented with the Huber loss and trained AerialMPTNet with it. Table 6.8 shows a comparison of AerialMPTNet trained with L1 and Huber Loss. Similar to the previous experiments, the results for the KIT AIS datasets indicate that the L1 loss works

Table 6.8: Comparison of AerialMPTNet trained with L1 and Huber Loss.

Dataset	Loss	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS pedestrian	L1	**40.6**	**39.7**	**41.5**	**45.1**	**43.2**	**103.45**	1043	**28.1**	55.3	**16.6**	**7138**	**6597**	236	897	**-16.2**	**69.6**	**-14.2**
KIT AIS pedestrian	Huber	38.8	37.9	39.7	43.1	41.1	107.42	1043	25.0	**56.5**	18.5	7412	6845	**212**	**866**	-20.3	69.4	-18.6
AerialMPT	L1	37.8	36.5	39.3	**43.1**	**40.0**	**115.48**	1396	15.3	**49.9**	**34.8**	**14782**	**13022**	436	1269	**-23.4**	69.7	**-21.5**
AerialMPT	Huber	**38.0**	**36.7**	**39.5**	43.0	39.9	115.70	1396	**15.6**	48.4	36.0	14809	13051	**415**	**1196**	-23.5	**69.9**	-21.7
KIT AIS vehicle	L1	**70.0**	**68.3**	**71.8**	**73.9**	**70.3**	**14.41**	230	66.5	**20.9**	**12.6**	**1556**	**1299**	**29**	67	**42.0**	**76.3**	**42.6**
KIT AIS vehicle	Huber	67.2	65.5	69.0	70.6	67.1	15.98	230	**67.0**	17.4	15.6	1726	1461	34	**65**	35.2	76.1	35.9

Table 6.9: Comparison of AerialMPTNet and AerialMPTNet$_{OHEM}$.

Dataset	OHEM	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KIT AIS pedestrian	No	**40.6**	**39.7**	**41.5**	**45.1**	**43.2**	**103.45**	1043	**28.1**	55.3	**16.6**	**7138**	**6597**	**236**	897	**-16.2**	**69.6**	**-14.2**
KIT AIS pedestrian	Yes	38.6	37.7	39.4	42.7	40.9	107.75	1043	26.1	**55.8**	18.1	7435	6889	254	**854**	-21.2	69.5	-19.1
AerialMPT	No	**37.8**	**36.5**	**39.3**	**43.1**	**40.0**	**115.48**	1396	15.3	**49.9**	**34.8**	**14782**	**13022**	436	**1269**	**-23.4**	69.7	**-21.5**
AerialMPT	Yes	37.2	35.8	38.7	42.4	39.3	117.31	1396	**16.0**	46.8	37.2	15016	13181	**430**	1284	-25.1	**69.8**	-23.2
KIT AIS vehicle	No	70.0	68.3	71.8	73.9	70.3	14.41	230	66.5	**20.9**	**12.6**	1556	1299	29	67	42.0	**76.3**	42.6
KIT AIS vehicle	Yes	**71.7**	**70.0**	**73.4**	**74.6**	**71.2**	**13.94**	230	**67.0**	19.6	13.4	**1505**	**1262**	**27**	**66**	**43.8**	75.5	**44.3**

better. The IDF1, IDP, IDR, Rcll, Prcn, FAR, ML, FP, FN, MOTA, MP, and MOTAL scores of AerialMPTNet trained with the L1 loss all lay ahead compared to the model trained with Huber loss. For the pedestrian set, MOTA is -20.3 for the Huber model, and -16.2 for the one trained with L1 loss. For the vehicle dataset, the difference is more considerable with the MOTA at 42.0 with L1 loss and 35.2 with Huber loss.

Again, the results for AerialMPTNet are more challenging to interpret. Huber loss improves the results, including IDF1, IDP, IDR, MT, ID, FM, and MOTP. However, the L1 loss shows more promising results for Rcll, Prcn, FAR, PT, ML, FP, FN, MOTA, and MOTAL. Nevertheless, we think that the L1 loss is the better choice to train regression trackers.

Last but not least, we experimented with OHEM and visualized the quantitative results in Table 6.9. Results on both pedestrian datasets are decreasing, with the MOTA falling from -16.2 to -21.2 and from -23.4 to -25.1 for the KIT AIS pedestrian and the AerialMPT dataset, respectively. For the KIT AIS vehicle dataset, however, results show a gradual improvement, MOTA climbs from 42.0 to 43.8 and the percentage of mostly tracked objects from 66.5 % to 67.90 %.

We argue that pedestrian movement is highly complex, and hence, the multiple passing of a similar situation to the tracker does not improve the results. For vehicles, however, which move along predetermined paths, OHEM improves the tracker's performance and indicates the beneficing for MOT. Further experiments have to prove this behavior on different datasets.

6.5 Comparison with other Methods

In this section, we compare the results of AerialMPTNet with a set of traditional methods, including MOSSE, KCF, Median Flow and CSRT, and DL-based approaches such as Tracktor++, Stacked DCFNet, and SMSOT-CNN.

Table 6.10 shows the quantitative results of different tracking methods on the KIT AIS pedestrian and AerialMPT datasets. In general, the DL-based methods outperform the traditional ones, with MOTA scores varying between -16.2 and -48.8 versus between -55.9 and-85.8, respectively. The percentage of mostly tracked and most lost objects varies between

0.8 % and 9.6 %, and between 36.5 % and 78.3 % for the traditional methods. Furthermore, CSRT is the best performing traditional method on both datasets based on MOTA (-55.9 and -64.6). It tracks 9.6 % and 2.9 % of the pedestrians mostly in the KIT AIS pedestrian and AerialMPT datasets, while it mostly loses 39.4 % and 59.3 %, respectively. Compared to the DL-based methods, the MT and ML scores vary between 0.1 % and 28.0 %, and between 16.6 % and 92.3 %. The deficient (0.1 %) and unusually high (92.3 %) scores are caused by Tracktor++, not working correctly with such small objects. AerialMPTNet outperforms all other methods on both datasets by the significantly highest MOTA (-16.2 and -23.4) and competitive MOTP (69.6 and 69.7) values. It mostly tracks 28.1 % and 15.3 % of the pedestrians while mostly losing only 16.6 % and 34.8 % of them. Among the previous DL-based methods, SMSOT-CNN achieves the most promising results on both datasets (MOTA: -35.0 and -37.2; MOTP: 70.0 and 68.0). Stacked DCFNet, which we adapted to handle multi-object tracking, outperforms SMSOT-CNN in terms of MOTP by 1.6 and 4.3 points, its MOTA values fall behind by 2.4 and 4.6 points. Tracktor++ performs the worst among the other DL-based methods due to suffering from a high amount of FNs and ID switches. Additionally, adding the LSTM module to the SNN module improves the MOTA by 17.2 and 9.1 points on the KIT AIS and AerialMPT datasets compared to SMSOT-CNN, respectively. Furthermore, adding the GCNN module to the SNN module improves the MOTA by 12.0 and 11.8 points. According to the table, considering both modules increases the MOTA by 18.8 and 13.8 points compared to SMSOT-CNN.

Table 6.11 shows the quantitative results of all trackers on the KIT AIS vehicle dataset. The DL-based methods outperform KCF, Median Flow, and MOSSE significantly, with MOTA scores between 37.1 and 46.6 versus -48.7 and -21.4. The percentage of mostly tracked and mostly lost objects lies between 19.6 % and 32.2 % and between 50.4 % and 27.8 % for KCF, MOSSE, and Median Flow. The DL-based methods are ranked better with the ratio of mostly tracked vehicles varying between 30.0 % and 69.1 % and the ratio of mostly lost objects between 22.6 % and 12.6 %. AerialMPTNet achieves MOTA and MOTP scores of 42.0 and 76.3. It mostly tracks 66.5 % of the vehicles, while it mostly loses 12.6 %. However, AerialMPTNet$_{LSTM}$ and AerialMPTNet$_{GCNN}$ outperform AerialMPTNet itself in terms of MOTA by 1.3 and 0.8 points (43.3 and 42.8). Among the DL-based methods, Stacked DCFNet performs best in terms of MOTA and MOTP, outperforming AerialMPTNet by 4.6 and 5.7 points, respectively. It tracks 69.1 % of the vehicle mostly while it loses 15.7 % of them mostly. Compared to the SMSOT-CNN baseline, MOTA and MOTP improve by 4.9 and 0.5 points. Furthermore, the performance of Tracktor++ increases significantly compared to the pedestrian scenarios. The main reason is the increased capability of the object detector. The detection of vehicles is significantly easier than the detection of pedestrians. Tracktor++ achieves a competitive MOTA of 37.1 without any ground truth initializations. The best method in terms of MOTA is CSRT. It outperforms all other methods with a MOTA of 51.1 and MOTP of 80.7. Compared to AerialMPTNet, the metrics improved by 10.1 and 5.7 points.

Additionally, we show all tracker rankings based on MOTA and MOTP for the KIT AIS pedestrian and AerialMPT dataset in Figure 6.17, and the rankings for the KIT AIS vehicle

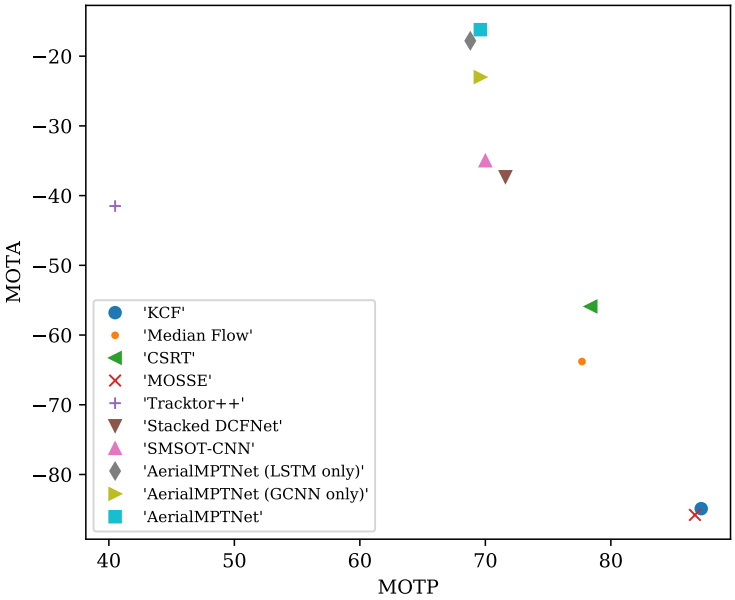
Table 6.10: Total Results of Different Trackers on the Pedestrian Datasets

Tracker	Dataset	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KCF	KIT AIS	9.0	8.8	9.3	10.3	9.8	165.56	1043	1.1	53.8	45.1	11426	10782	32	116	-84.9	**87.2**	-84.7
Median Flow	KIT AIS	18.5	18.3	18.8	19.5	19.0	144.72	1043	7.7	55.8	36.5	9986	9678	30	161	-63.8	77.7	-63.5
CSRT	KIT AIS	16.0	16.9	15.2	17.5	19.4	126.55	1043	9.6	51.0	39.4	8732	9924	91	254	-55.9	78.4	-55.1
MOSSE	KIT AIS	9.1	8.9	9.3	10.5	10.0	163.81	1043	0.8	54.0	45.2	11303	10765	31	133	-85.8	86.7	-83.5
Tracktor++	KIT AIS	6.6	9.0	5.2	10.8	18.7	**81.86**	1043	1.1	28.4	70.5	**5648**	10723	648	367	-41.5	40.5	–
Stacked DCFNet	KIT AIS	30.0	30.2	30.9	33.1	32.3	120.52	1043	13.8	62.6	23.6	8316	8051	139	651	-37.3	71.6	-36.1
SMSOT-CNN	KIT AIS	32.5	31.7	33.4	35.7	33.9	121.32	1043	22.2	56.0	21.8	8371	7730	**135**	**585**	-35.0	70.0	-33.9
AerialMPTNet$_{LSTM}$ (Ours)	KIT AIS	39.7	38.8	40.6	44.6	42.6	104.78	1043	**28.9**	53.8	17.3	7230	6661	270	886	-17.8	68.8	-15.5
AerialMPTNet$_{GCNN}$ (Ours)	KIT AIS	37.5	36.7	38.4	42.0	40.0	109.49	1043	25.3	**55.3**	19.4	7555	6980	259	814	-23.0	69.6	-20.9
AerialMPTNet (Ours)	KIT AIS	**40.6**	**39.7**	**41.5**	**45.1**	**43.2**	103.45	1043	28.1	**55.3**	**16.6**	7138	**6597**	236	897	**-16.2**	69.6	**-14.2**
KCF	AerialMPT	11.9	11.5	12.3	13.4	12.5	167.24	1396	3.7	17.0	79.3	21407	19820	86	212	-80.5	77.2	-80.1
Median Flow	AerialMPT	12.2	12.0	12.4	13.1	12.7	161.97	1396	1.7	20.2	78.1	20732	19883	46	144	-77.7	77.8	-77.5
CSRT	AerialMPT	16.9	16.6	17.1	20.3	19.7	148.52	1396	2.9	37.8	59.3	19011	18235	426	668	-64.6	74.6	-62.7
MOSSE	AerialMPT	12.1	11.7	12.4	13.7	12.9	165.66	1396	3.8	17.9	78.3	21204	19749	85	194	-79.3	**80.0**	-78.9
Tracktor++	AerialMPT	4.0	8.8	3.1	5.0	8.7	**93.02**	1396	0.1	7.6	92.3	**11907**	21752	399	345	-48.8	40.3	–
Stacked DCFNet	AerialMPT	28.0	27.6	28.5	31.4	30.4	128.30	1396	131	617	648	16422	15712	322	944	-41.8	72.3	-40.4
SMSOT-CNN	AerialMPT	32.0	30.7	33.4	36.6	33.6	129.13	1396	10.7	47.7	41.6	16529	14515	359	1082	-37.2	68.0	-35.6
AerialMPTNet$_{LSTM}$ (Ours)	AerialMPT	35.7	34.5	37.0	40.5	37.7	119.40	1396	12.8	49.8	37.4	15283	13627	409	1376	-28.1	70.1	-26.3
AerialMPTNet$_{GCNN}$(Ours)	AerialMPT	37.0	35.7	38.3	42.0	39.1	117.05	1396	**15.6**	46.0	38.4	14983	13279	433	1229	-25.4	69.7	-23.5
AerialMPTNet (Ours)	AerialMPT	**37.8**	**36.5**	**39.3**	**43.1**	**40.0**	115.48	1396	15.3	**49.9**	**34.8**	14782	**13022**	436	1269	**-23.4**	69.7	**-21.5**

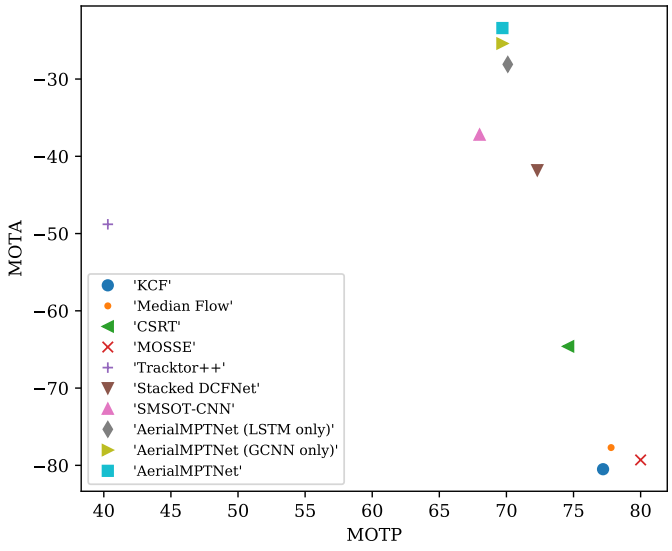
Table 6.11: Total Results of Different Trackers on the KIT AIS Vehicle Dataset.

Tracker	IDF1↑	IDP↑	IDR↑	Rcll↑	Prcn↑	FAR↓	GT	MT%↑	PT%↑	ML%↓	FP↓	FN↓	ID↓	FM↓	MOTA↑	MOTP↑	MOTAL↑
KCF	41.3	39.0	43.9	45.6	40.4	30.92	230	27.0	33.5	39.5	3339	2708	53	96	-22.6	72.3	-21.6
Median Flow	42.0	39.5	44.9	46.3	40.8	31.00	230	32.2	40.0	27.8	3348	2669	23	47	-21.4	**82.0**	-21.0
CSRT	**76.7**	**72.1**	**81.9**	**83.1**	73.1	14.07	230	**72.6**	21.7	**5.7**	1520	**841**	21	46	**52.1**	80.7	**52.5**
MOSSE	29.0	27.4	30.8	32.4	28.8	36.82	230	19.6	30.0	50.4	3977	3364	56	81	-48.7	75.0	-47.6
Tracktor++	55.3	66.6	47.2	57.3	**80.7**	**6.31**	230	30.0	**47.4**	22.6	**681**	2125	323	204	37.1	77.4	–
Stacked DCFNet	73.8	71.2	76.6	77.2	71.8	14.00	230	69.1	15.2	15.7	1512	1133	**9**	**39**	46.6	**82.0**	46.8
SMSOT-CNN	68.0	66.4	69.7	71.3	67.9	15.53	230	65.7	20.4	13.9	1677	1426	27	80	37.1	75.8	37.6
AerialMPTNet$_{LSTM}$ (Ours)	71.6	69.8	73.4	74.5	70.9	14.11	230	67.4	19.6	13.0	1524	1267	30	60	43.3	75.7	43.9
AerialMPTNet$_{GCNN}$ (Ours)	71.1	69.4	72.9	74.1	70.6	14.22	230	67.0	18.7	14.3	1536	1289	22	58	42.8	75.9	43.2
AerialMPTNet (Ours)	70.0	68.3	71.8	73.9	70.3	14.41	230	66.5	20.9	12.6	1556	1299	29	67	42.0	76.3	42.6

dataset in Figure 6.18. The diagrams offer an intuitive way to illustrate the rankings clearly. AerialMPTNet ranks best in terms of MOTA for both pedestrian dataset while achieving competitive MOTP. Median Flow, for example, achieves a very high MOTP score; however, the reason for this behavior is the low number of matched tracklet-object pairs beyond the first frame since the tracker is not able to track many objects. Hence, the MOTP score solely is not a good performance indicator. For the KIT AIS vehicle dataset, AerialMPTNet shows worse performance than other methods based on MOTA and MOTP. CSRT and Stacked DCFNet perform favorably for vehicle tracking.



(a) KIT AIS pedestrian



(b) AerialMPT

Figure 6.17: Tracker rankings based on MOTA and MOTP for the KIT AIS pedestrian and the AerialMPT dataset. AerialMPTNet ranks best in terms of MOTA and achieves a competitive MOTP for both datasets.

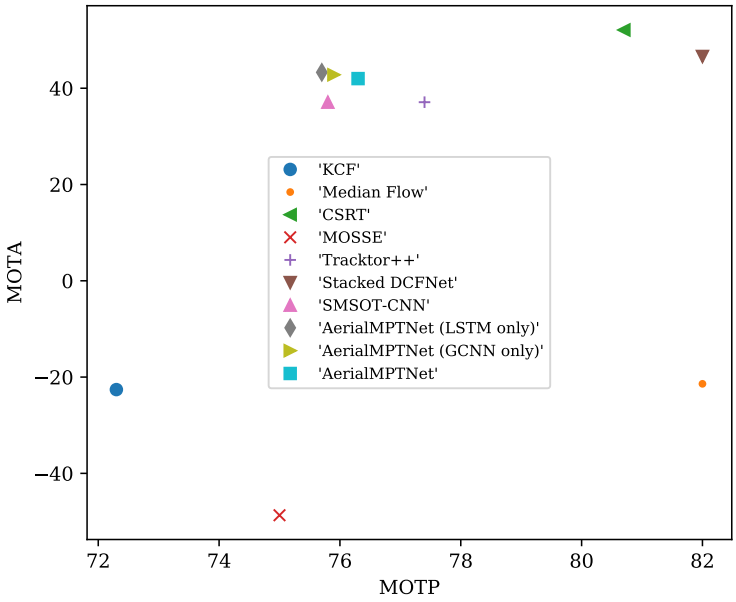


Figure 6.18: Tracker rankings based on MOTA and MOTP for the KIT AIS vehicle dataset.

7 Conclusion & Future Work

In this master thesis, we proposed AerialMPTNet, a DNN for pedestrian- and vehicle-tracking in aerial imagery. AerialMPTNet is a regression-based multi-object tracker and takes two image crops (previous and current) for each target as input. Multiple objects are given to the network as a mini-batch. The object position is known and has to be determined, respectively. AerialMPTNet consists of three different subnetworks, namely an SNN, an LSTM, and a GraphCNN and fuses appearance, temporal, and graphical features. The SNN takes both crops as input and extracts important appearance features that are used by 4 linear layers to regress the top-left and bottom-right corner of the target's bounding box in the current crop. The SNN-based method is the standard procedure in most regression-based trackers. However, such networks typically lack important target information such as previous movements and the relationship between adjacent objects.

We deal with these issues by using the LSTM and the GraphCNN structures and give the final output of the network as input to the LSTM module, which learns to predict the next movement of the target. The produced features are concatenated with the SNN's features and given to the linear layers in the next iteration. Previous work often employs Kalman filters or linear motion models to estimate target positions in consecutive frames. However, the initialization with matching hyperparameters can be difficult in such cases. In contrast to these methods, we encode the motion model in an end-to-end fashion and learn characteristics based on the data itself. Furthermore, previous work on regression-based tracking and motion models use the estimates to shift the search windows accordingly, but we directly estimate the object position inside of the crop without moving it.

Additionally, we save the position estimates for all targets and create a directed neighbor graph for each object. The graph consists of the object position itself, and we extend it by the vectors to its 9 closest neighbors. Afterwards, we give these graphs to the GraphCNN, which learns to encode the relationship between adjacent objects. Together with the LSTM module's features, we concatenate the GraphCNN features with the SNN's features. Similar to the LSTM module, we encode these relationships in an end-to-end way.

We evaluate AerialMPTNet on the KIT AIS and the AerialMPT dataset. The KIT AIS dataset composes two branches, a pedestrian and a vehicle set. As of today, AerialMPT only consists of a pedestrian dataset; however, the authors aim to create a vehicle dataset as well. The results on the KIT AIS pedestrian and the AerialMPT dataset indicate that our model successfully tackles the challenge of tracking small objects in aerial imagery, leading to a MOTA improvement by 18.2 and 13.8 points compared to the SMSOT-CNN baseline while holding a competitive MOTP score of 69.6 and 69.7. We confirm this finding with several visual inspections and show that ID switches and loosing of tracks during object crossings and crowded situations improve significantly compared to the baseline. The results on the

KIT AIS vehicle branch also improve compared to the SMSOT-CNN baseline: MOTA score increases by 4.9, reaching a final MOTA of 42.0 points. We reach a competitive MOTP of 76.3.

Nevertheless, the total MOTA scores are negative. The general performance indicates that there is room for improvement. Results also show that AerialMPTNet loses objects that move out of the search window. Different object speeds and GSDs cause this problem.

Additionally, we performed an intensive comparison with several traditional and DNN-based tracking methods, including KCF, Median Flow, MOSSE, CSRT, Tracktor++, and Stacked DCFNet. AerialMPTNet outperforms all these previous works by a large margin for the pedestrian datasets. However, it falls behind CSRT and Stacked DCFNet for vehicle tracking.

Furthermore, we studied the usage of OHEM, Squeeze-and-Excitation layers, and Huber Loss for regression-based trackers. For the vehicle part, OHEM shows promising results. It brings 1.8 point improvement in terms of MOTA. However, for the pedestrian datasets, the result did not improve. The results for the Squeeze-and-Excitation layers were also not conclusive. For both KIT AIS datasets, results worsened. However, the results for AerialMPT show significant improvements, with the MOTA improving from -23.4 to -21.4. We argue that the reason for this behavior is the different quality of the datasets. However, more experiments are needed to confirm this finding. The usage of Huber loss does not improve the tracking performance, confirmed on all datasets.

There are several aspects which we consider for future work. Within our framework, the size of the search window is dependent on the size of the object's bounding box only. Nevertheless, there exist multiple factors influencing the movement range of an object, such as object speed and speed deviation, as well as the GSD. We leave the development of an adaptive search window size as an essential step to reduce the number of lost tracks for future work. Additionally, the SNN module is based initially on the GOTURN's implementation and could be replaced by a newer SNN module. We think that this change would improve the results of feature extraction significantly. As of today, the training process of most DNN-based tracking methods relies on specific loss functions, which do not correlate with tracking evaluation metrics such as MOTA and MOTP. The main reason for this choice is that most tracking measures are not differentiable. This finding is also valid for this master thesis, where we experimented with the L1 and the Huber loss. However, Xu et al. [100] recently proposed a framework capable of tackling this challenge by using differentiable proxies of MOTA and MOTP, which they combine in a loss function and can be used to optimize deep trackers directly. Based on the results of Xu et al., we suggest using this framework also to improve the tracking performance of our tracker.

List of Figures

List of Tables

Acronyms

ADI Accumulative Difference Image.

AerialMPT Aerial Multi-Pedestrian Tracking.

CNN Convolutional Neural Network.

DCF Discriminative Correlation Filter.

DL Deep Learning.

DLR German Aerospace Center.

DLR-ACD DLR's Aerial Crowd Dataset.

DNN Deep Neural Networks.

EOT Euclidean Online Tracking.

FPS Frames per Second.

GAN Generative Adversarial Network.

GCNN Graph Convolutional Neural Network.

GSD Ground Sampling Distance.

HOG Histogram of Gradients.

IoU Intersection over Union.

KCF Kernelized Correlation Filter.

LRN Local Response Normalization.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

ML Mostly Lost.

MOSSE Minimum Output Sum of Squared Errors.

MOT Multi-Object Tracker.

MOTA Multiple-Object Tracker Accuracy.

MOTP Multiple-Object Tracker Precision.

MSE Mean Squared Error.

MT Mostly Tracked.

OCR Optical Character Recognition.

OHEM Online Hard Example Mining.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

SE Squeeze-and-Excitation.

SIFT Scale-Invariant Feature Transform.

SMSOT-CNN Stack of Multiple Single Object Tracking CNNs.

SNN Siamese Neural Network.

SOT Single-Object Tracker.

TUM Technical University of Munich.

VOT Visual Object Tracking.

Bibliography

[1] N. Wojke, A. Bewley, and D. Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[2] P. Bergmann, T. Meinhardt, and L. Leal-Taixe. "Tracking without bells and whistles". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 941–951.

[3] Y. Xiang, A. Alahi, and S. Savarese. "Learning to track: Online multi-object tracking by decision making". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4705–4713.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. "Fully-convolutional siamese networks for object tracking". In: *European conference on computer vision*. Springer. 2016, pp. 850–865.

[5] E. V. Cuevas, D. Zaldivar, and R. Rojas. "Kalman filter for vision tracking". In: (2005).

[6] E. Cuevas, D. Zaldivar, and R. Rojas. "Particle filter in vision tracking". In: *e-Gnosis* 5 (2007), pp. 1–11.

[7] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. "Visual object tracking using adaptive correlation filters". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 2544–2550.

[8] G. Boudoukh, I. Leichter, and E. Rivlin. "Visual tracking of object silhouettes". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2009, pp. 3625–3628.

[9] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[10] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei. "Deep Learning for Visual Tracking: A Comprehensive Survey". In: *arXiv preprint arXiv:1912.00535* (2019).

[11] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[13] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[14] L. Wang, W. Ouyang, X. Wang, and H. Lu. "Visual tracking with fully convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3119–3127.

[15] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang. "Robust visual tracking via convolutional networks without training". In: *IEEE Transactions on Image Processing* 25.4 (2016), pp. 1779–1792.

[16] H.-I. Kim and R.-H. Park. "Residual LSTM attention network for object tracking". In: *Ieee Signal Processing Letters* 25.7 (2018), pp. 1029–1033.

[17] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. "High performance visual tracking with siamese region proposal network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8971–8980.

[18] D. Held, S. Thrun, and S. Savarese. "Learning to Track at 100 FPS with Deep Regression Networks". en. In: *arXiv:1604.01802 [cs]* (Aug. 2016). arXiv: 1604.01802. URL: http://arxiv.org/abs/1604.01802 (visited on 11/04/2019).

[19] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang. "Vital: Visual tracking via adversarial learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8990–8999.

[20] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang. "Deep reinforcement learning for visual object tracking in videos". In: *arXiv preprint arXiv:1701.08936* (2017).

[21] G. Pingali, A. Opalach, and Y. Jean. "Ball tracking and virtual replays for innovative tennis broadcasts". In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 4. IEEE. 2000, pp. 152–156.

[22] X. Zhang and S. Payandeh. "Application of visual tracking for robot-assisted laparoscopic surgery". In: *Journal of Robotic systems* 19.7 (2002), pp. 315–328.

[23] T. M. Wood, C. A. Yates, D. A. Wilkinson, and G. Rosser. "Simplified multitarget tracking using the PHD filter for microscopic video data". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.5 (2011), pp. 702–713.

[24] 1. C. U.S. House Hearing. *Remote Sensing Data: Applications and Benefits*. Tech. rep. Serial No. 110-91, retrieved January 2, 2020: https://www.govinfo.gov/content/pkg/CHRG-110hhrg41573/html/CHRG-110hhrg41573.html. Subcommittee on Space, Aeronautics, Committee on Science, and Technology, Apr. 2008.

[25] J. Everaerts et al. "The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 37.2008 (2008), pp. 1187–1192.

[26] H. Heier. "Applications and market for digital airborne cameras". In: *Photogrammetric Week 1999* (1999), pp. 43–49.

[27] V. Klemas. "Remote sensing techniques for studying coastal ecosystems: An overview". In: *Journal of Coastal Research* 27.1 (2011), pp. 2–17.

[28] V. Reilly, H. Idrees, and M. Shah. "Detection and tracking of large number of targets in wide area surveillance". In: *European conference on computer vision*. Springer. 2010, pp. 186–199.

[29] L. Meng and J. P. Kerekes. "Object tracking using high resolution satellite imagery". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 5.1 (2012), pp. 146–152.

[30] R. Bahmanyar, S. Azimi, and P. Reinartz. "MULTIPLE VEHICLES AND PEOPLE TRACKING IN AERIAL IMAGERY USING STACK OF MICRO SINGLE-OBJECT-TRACKING CNNS". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2019), pp. 163–170.

[31] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. "MOT16: A benchmark for multi-object tracking". In: *arXiv preprint arXiv:1603.00831* (2016).

[32] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang. "Learning dynamic siamese network for visual object tracking". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1763–1771.

[33] A. He, C. Luo, X. Tian, and W. Zeng. "A twofold siamese network for real-time object tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4834–4843.

[34] S. J. Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[35] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[37] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[38] E. C. Barrett. *Introduction to environmental remote sensing*. Routledge, 2013.

[39] A. Dey. "Machine learning algorithms: a review". In: *International Journal of Computer Science and Information Technologies* 7.3 (2016), pp. 1174–1179.

[40] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[41] Y. LeCun and C. Cortes. "MNIST handwritten digit database". In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.

[42] A. Krizhevsky, G. Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[43] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[44] D. Harrison Jr and D. L. Rubinfeld. "Hedonic housing prices and the demand for clean air". In: *Journal of environmental economics and management* 5.1 (1978), pp. 81–102.

[45] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. "Modeling wine preferences by data mining from physicochemical properties". In: *Decision Support Systems* 47.4 (2009), pp. 547–553.

[46] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[47] R. Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[48] K. O'Shea and R. Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[49] J. Hu, L. Shen, and G. Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[50] M. Lin, Q. Chen, and S. Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[51] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[52] A. L. Maas, A. Y. Hannun, and A. Y. Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[53] P. Ramachandran, B. Zoph, and Q. V. Le. "Swish: a self-gated activation function". In: *arXiv preprint arXiv:1710.05941* 7 (2017).

[54] P. J. Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[55] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung. "Handcrafted and deep trackers: Recent visual object tracking approaches and trends". In: *ACM Computing Surveys (CSUR)* 52.2 (2019), pp. 1–44.

[56] R. E. Kalman. "A new approach to linear filtering and prediction problems". In: (1960).

[57] L. M. A. Y. C. F. S. A. G. S. K. P. S. Godsilll. "Overview of Bayesian sequential Monte Carlo methods for group and extended object tracking". In: (2014).

[58] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu. "DCFNet: Discriminant Correlation Filters Network for Visual Tracking". en. In: *arXiv:1704.04057 [cs]* (Apr. 2017). arXiv: 1704.04057. URL: http://arxiv.org/abs/1704.04057 (visited on 11/07/2019).

[59] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. "Hierarchical convolutional features for visual tracking". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3074–3082.

[60] N. Wojke, A. Bewley, and D. Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE International Conference on Image Processing (ICIP)*. ISSN: 2381-8549. Sept. 2017, pp. 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[61] C. Huang, B. Wu, and R. Nevatia. "Robust object tracking by hierarchical association of detection responses". In: *European Conference on Computer Vision*. Springer. 2008, pp. 788–801.

[62] X. Lu, C. Ma, B. Ni, X. Yang, I. Reid, and M.-H. Yang. "Deep Regression Tracking with Shrinkage Loss". en. In: (), p. 17.

[63] L. Wang, W. Ouyang, X. Wang, and H. Lu. "Stct: Sequentially training convolutional networks for visual tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1373–1381.

[64] C. Huang, S. Lucey, and D. Ramanan. "Learning policies for adaptive tracking with deep feature cascades". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 105–114.

[65] D. Chahyati, M. I. Fanany, and A. M. Arymurthy. "Tracking people by detection using CNN features". In: *Procedia Computer Science* 124 (2017), pp. 167–172.

[66] Y. Zhang, J. Wang, and X. Yang. "Real-time vehicle detection and tracking in video based on faster R-CNN". In: *Journal of Physics: Conference Series*. Vol. 887. 1. IOP Publishing. 2017, p. 012068.

[67] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. "A boosted particle filter: Multitarget detection and tracking". In: *European conference on computer vision*. Springer. 2004, pp. 28–39.

[68] R. Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[69] G. D. Hager and P. N. Belhumeur. "Real-time tracking of image regions with changes in geometry and illumination". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 1996, pp. 403–410.

[70] K. Briechle and U. D. Hanebeck. "Template matching using fast normalized cross correlation". In: *Optical Pattern Recognition XII*. Vol. 4387. International Society for Optics and Photonics. 2001, pp. 95–102.

[71] S. Avidan. "Ensemble tracking". In: *IEEE transactions on pattern analysis and machine intelligence* 29.2 (2007), pp. 261–271.

[72] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. "Struck: Structured output tracking with kernels". In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2015), pp. 2096–2109.

[73] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. "High-speed tracking with kernelized correlation filters". In: *IEEE transactions on pattern analysis and machine intelligence* 37.3 (2014), pp. 583–596.

[74] A. Sadeghian, A. Alahi, and S. Savarese. "Tracking the untrackable: Learning to track multiple cues with long-term dependencies". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 300–311.

[75] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[76] Z. Kalal, K. Mikolajczyk, and J. Matas. "Forward-backward error: Automatic detection of tracking failures". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 2756–2759.

[77] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan. "Discriminative correlation filter with channel and spatial reliability". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6309–6318.

[78] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)*. ISSN: 2381-8549. Sept. 2016, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

[79] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[80] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian. "Mars: A video benchmark for large-scale person re-identification". In: *European Conference on Computer Vision*. Springer. 2016, pp. 868–884.

[81] A. Jadhav, P. Mukherjee, V. Kaushik, and B. Lall. "Aerial multi-object tracking by detection using deep association networks". en. In: *arXiv:1909.01547 [cs]* (Sept. 2019). arXiv: 1909.01547. URL: http://arxiv.org/abs/1909.01547 (visited on 11/05/2019).

[82] C. Benedek, T. Szirányi, Z. Kato, and J. Zerubia. "Detection of object motion regions in aerial image pairs with a multilayer Markovian model". In: *IEEE Transactions on Image Processing* 18.10 (2009), pp. 2303–2315.

[83] M. Butenuth, F. Burkert, F. Schmidt, S. Hinz, D. Hartmann, A. Kneidl, A. Borrmann, and B. Sirmacek. "Integrating pedestrian simulation, tracking and event detection for crowd analysis". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE. 2011, pp. 150–157.

[84] F. Schmidt and S. Hinz. "A Scheme for the Detection and Tracking of People Tuned for Aerial Image Sequences". In: *Photogrammetric Image Analysis (PIA)*. Ed. by U. Stilla, F. Rottensteiner, H. Mayer, B. Jutzi, and M. Butenuth. LNCS 6952. ISPRS. Munich, Germany: Springer, Heidelberg, Oct. 2011, pp. 257–270. DOI: 10.1007/978-3-642-24393-6_22.

[85] K. Liu and G. Mattyus. "Fast multiclass vehicle detection on aerial images". In: *IEEE Geoscience and Remote Sensing Letters* 12.9 (2015), pp. 1938–1942.

[86] S. Qi, J. Ma, J. Lin, Y. Li, and J. Tian. "Unsupervised ship detection based on saliency and S-HOG descriptor from optical satellite images". In: *IEEE Geoscience and Remote Sensing Letters* 12.7 (2015), pp. 1451–1455.

[87] M. Yokoyama and T. Poggio. "A contour-based moving object detection and tracking". In: *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. IEEE. 2005, pp. 271–276.

[88] R. Bahmanyar, E. Vig, and P. Reinartz. "MRCNet: Crowd Counting and Density Map Estimation in Aerial and Ground Imagery". In: *arXiv preprint arXiv:1909.12743* (2019).

[89] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. "Performance measures and a data set for multi-target, multi-camera tracking". In: *European Conference on Computer Vision*. Springer. 2016, pp. 17–35.

[90] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu. "Dcfnet: Discriminant correlation filters network for visual tracking". In: *arXiv preprint arXiv:1704.04057* (2017).

[91] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[92] R. Rastogi, I. Thaniarasu, and S. Chandra. "Design implications of walking speed for pedestrian facilities". In: *Journal of transportation engineering* 137.10 (2011), pp. 687–696.

[93] K. Finnis and D. Walton. "Field observations of factors influencing walking speeds". In: *Ergonomics* (2006).

[94] D. L. Strayer and F. A. Drew. "Profiles in driver distraction: Effects of cell phone conversations on younger and older drivers". In: *Human factors* 46.4 (2004), pp. 640–649.

[95] H. Rakha, I. El-Shawarby, and J. R. Setti. "Characterizing driver behavior on signalized intersection approaches at the onset of a yellow-phase trigger". In: *IEEE Transactions on Intelligent Transportation Systems* 8.4 (2007), pp. 630–640.

[96] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. "Social lstm: Human trajectory prediction in crowded spaces". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 961–971.

[97] H. Xue, D. Q. Huynh, and M. Reynolds. "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1186–1194.

[98] A. Vemula, K. Muelling, and J. Oh. "Social attention: Modeling attention in human crowds". In: *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–7.

[99] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[100] Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixé, and X. Alameda-Pineda. "How To Train Your Deep Multi-Object Tracker". In: *Computer Vision and Pattern Recognition*. 2020.